COMPUTABILITY IN EUROPE 2013

cie2013.disco.unimib.it

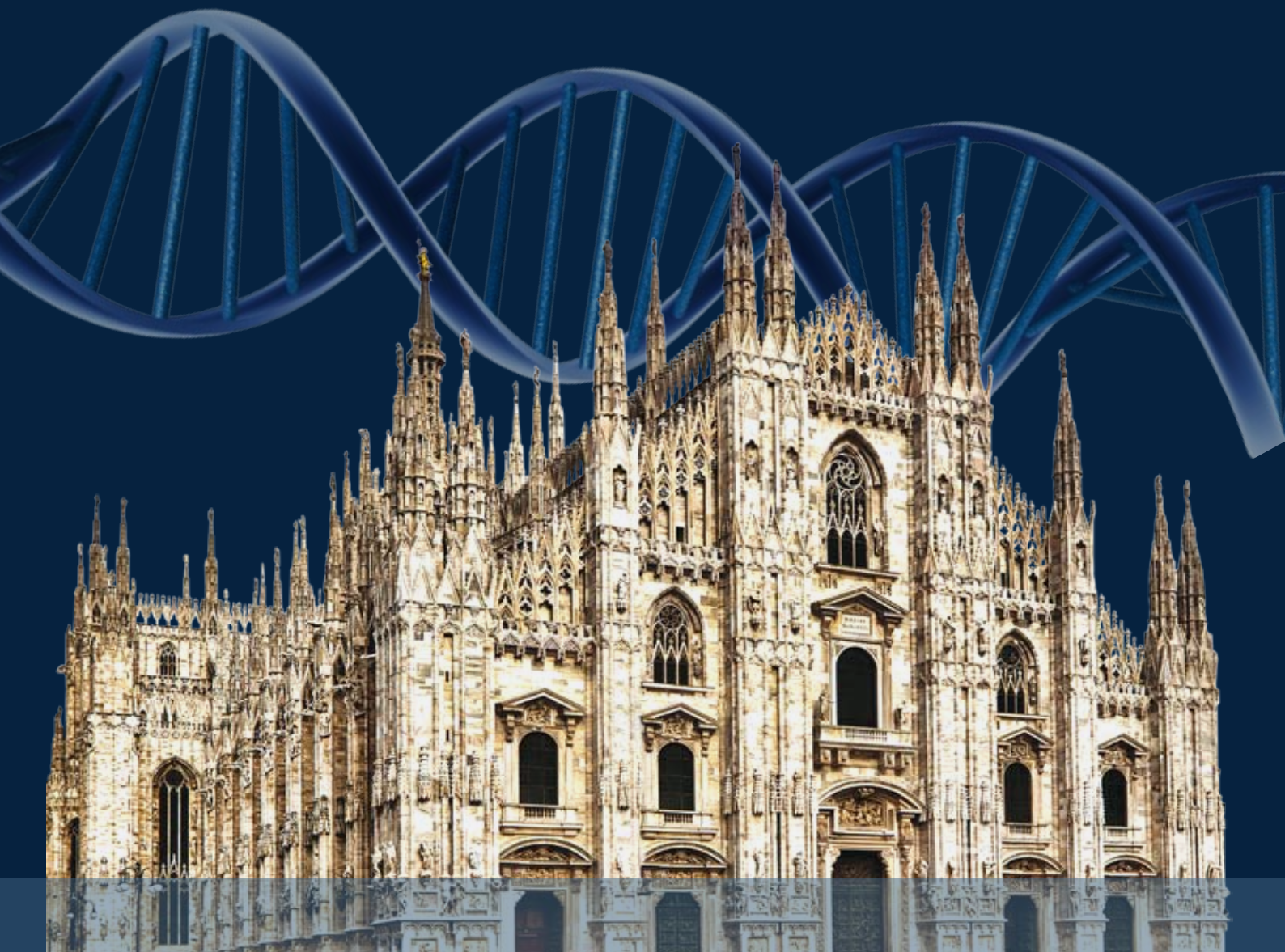
The Nature of Computation

Logic, Algorithms, Applications

1st July - 5th July

Università degli Studi di Milano-Bicocca, Milan, Italy

Informal Proceedings



Preface

Computability in Europe 2013 (CiE 2013) followed the *Turing Centenary Conference* CiE 2012 in celebrating the enormous influence of Turing's work on the specific focus of the CiE conference series: the development of a multi-disciplinary and modern view of computation and computability. The interest for computation in nature (which was also the motivation for Turing's work on biological pattern formation) as reflected in the title of CiE 2013, *The Nature of Computation*, connects biology and computer science and has given rise to modern disciplines of research as well as new perspectives on computation.

In particular, CiE 2013 was focused on the unexpected changes that studies on nature have brought to several areas of mathematics, physics, and computer science. Two complementary research perspectives pervade the *Nature of Computation* theme. One is focused on the understanding of new computational paradigms, inspired by processes occurring in the biological world and resulting in a deeper and modern understanding of the theory of computation. The other perspective is on our understanding of how computations really occur in nature, on how we can interact with these computations, and their applications.

CiE 2013 was the ninth meeting in the conference series *Computability in Europe* organized by the Association CiE. The association promotes the development of computability-related science, ranging from mathematics, computer science and applications in various natural and engineering sciences, such as physics and biology, as well as the promotion of related fields, such as philosophy and history of computing. In particular, the conference series successfully brings together the mathematical, logical and computer sciences communities that are interested in developing computability related topics. This year this scope was strengthened by the co-location of CiE 2013 with UCNC 2013 (*Unconventional Computation and Natural Computation*), with Giancarlo Mauri as the chair of the programme committee.

The two conferences, CiE 2013 and UCNC 2013, were held at the University of Milano-Bicocca in Milan, Italy. They shared one plenary invited talk, given by Endre Szemerédi (Budapest & Piscataway NJ), winner of the Abel Prize in 2012, and two tutorials, one by Grzegorz Rozenberg (Leiden & Boulder CO) and one by Gilles Brassard (Montréal QC). Moreover, some satellite events are organized around the two conferences.

The eight previous CiE conferences were held in Amsterdam (The Netherlands) in 2005, Swansea (Wales) in 2006, Siena (Italy) in 2007, Athens (Greece) in 2008, Heidelberg (Germany) in 2009, Ponta Delgada (Portugal) in 2010, Sofia (Bulgaria) in 2011, and Cambridge (England) in 2012. The proceedings of these meetings were all published in the Springer series *Lecture Notes in Computer Science*. The annual CiE conference has become a major event and is the largest international meeting focused on computability theoretic issues. The next meeting in 2014 will be held in Budapest (Hungary).

The series is coordinated by the CiE Conference Series Steering Committee consisting of Luís Antunes (Porto, Secretary), Arnold Beckmann (Swansea), Laurent Bienvenu (Paris), Natasha Jonoska (Tampa FL), Viv Kendon (Leeds),

Benedikt Löwe (Amsterdam & Hamburg, chair), Mariya Soskova (Sofia & Berkeley CA), and Peter van Emde Boas (Amsterdam).

Programme Committee

The Programme Committee of CiE 2013 was responsible for the selection of the invited speakers, the special session organizers and for running the reviewing process of all submitted regular contributions. It consisted of Gerard Alberts (Amsterdam), Luís Antunes (Porto), Arnold Beckmann (Swansea), Laurent Bienvenu (Paris), Paola Bonizzoni (Milan, co-chair), Vasco Brattka (Munich & Cape Town, co-chair), Cameron Buckner (Houston TX), Bruno Codenotti (Pisa), Stephen Cook (Toronto ON), Barry Cooper (Leeds), Ann Copestake (Cambridge), Erzsébet Csuhaj-Varjú (Budapest), Anuj Dawar (Cambridge), Gianluca Della Vedova (Milan), Liesbeth De Mol (Ghent), Jérôme Durand-Lose (Orléans), Viv Kendon (Leeds), Bjørn Kjos-Hanssen (Honolulu HI), Antonina Kolokolova (St. John's NF), Benedikt Löwe (Amsterdam & Hamburg), Giancarlo Mauri (Milan), Rolf Niedermeier (Berlin), Geoffrey Pullum (Providence RI & Edinburgh), Nicole Schweikardt (Frankfurt), Sonja Smets (Amsterdam), Susan Stepney (York), S.P. Suresh (Chennai), and Peter van Emde Boas (Amsterdam).

Structure and Programme of the Conference

The programme committee invited six speakers to give plenary lectures: Ulle Endriss (Amsterdam), Lance Fortnow (Atlanta GA), Anna Karlin (Seattle WA), Bernard Moret (Lausanne), Mariya Soskova (Sofia & Berkeley CA), and Endre Szemerédi (Budapest & Piscataway NJ; joint invitee of CiE 2013 and UCNC 2013).

These plenary speakers were invited to publish abstracts or papers in this volume. Karlin's lecture was the *2013 APAL Lecture* funded by Elsevier, Fortnow's lecture was the *2013 EACSL Lecture* funded by the *European Association for Computer Science Logic*, and Szemerédi's lecture was funded by the Department of Mathematics and its Applications of the University of Milano-Bicocca. In addition to the plenary lectures, the conference had two tutorials by Gilles Brassard (Montréal QC) and Grzegorz Rozenberg (Leiden & Boulder CO).

Springer-Verlag generously funded two awards that were given during the CiE 2013 conference. Nicolas de Rugy-Altherre was awarded the *Best Student Paper Award* for his paper "*Determinant versus Permanent: Salvation via Generalization?*" Shankara Narayanan Krishna, Marian Gheorghe, and Ciprian Dragomir were awarded the *Best Paper on Natural Computing Award* for their paper "*Some Classes of Generalised Communicating P Systems and Simple Kernel P Systems.*"

The conference CiE 2013 had six special sessions: two sessions *Computational Molecular Biology* and *Computation in Nature*, were devoted to the special focus of CiE 2013. In addition to this, new challenges arising in computations in the real world were faced in the session on *Data Streams and Compression.* The remaining three sessions were on *Algorithmic Randomness*, *Computational*

Complexity in the Continuous World, and *History of Computation*. Speakers in these special sessions were selected by the special session organizers and could contribute a paper to this volume.

Algorithmic Randomness.
Organizers. Mathieu Hoyrup (Nancy) and André Nies (Auckland).
Speakers. Johanna Franklin (Storrs CT), Noam Greenberg (Wellington), Joseph S. Miller (Madison WI), Nikolay Vereshchagin (Moscow).

Computational Complexity in the Continuous World.
Organizers. Akitoshi Kawamura (Tokyo) and Robert Rettinger (Hagen).
Speakers. Mark Braverman (Princeton NJ), Daniel S. Graça (Faro), Joris van der Hoeven (Palaiseau), Chee K. Yap (New York NY).

Computational Molecular Biology.
Organizers. Alessandra Carbone (Paris) and Jens Stoye (Bielefeld).
Speakers. Sebastian Böcker (Jena), Marília D.V. Braga (Duque de Caxias), Andrea Pagnani (Torino), Laxmi Parida (Yorktown Heights NY).

Computation in Nature.
Organizers. Mark Delay (London ON) and Natasha Jonoska (Tampa FL).
Speakers. Jerome Durand-Lose (Orléans), Giuditta Franco (Verona), Lila Kari (London ON), Darko Stefanovic (Albuquerque NM).

Data Streams and Compression.
Organizers. Paolo Ferragina (Pisa) and Andrew McGregor (Amherst MA).
Speakers. Graham Cormode (Florham Park NJ), Irene Finocchi (Rome), Andrew McGregor (Amherst MA), Marinella Sciortino (Palermo).

History of Computation.
Organizers. Gerard Alberts (Amsterdam) and Liesbeth De Mol (Ghent).
Speakers. David Alan Grier (Washington DC), Thomas Haigh (Milwaukee WI), Ulf Hashagen (Munich), Matti Tedre (Stockholm).

All authors who have contributed to this conference are encouraged to submit significantly extended versions of their papers with unpublished research content to *Computability. The Journal of the Association CiE*.

Organisation and Acknowledgements.

The conference CiE 2013 was organised by Stefano Beretta (Milan), Paola Bonizzoni (Milan), Gianluca Della Vedova (Milan), Alberto Dennunzio (Milan), Riccardo Dondi (Bergamo), Giancarlo Mauri (Milan), Yuri Pirola (Milan) and Raffaella Rizzi (Milan).

The Steering Committee of the conference series CiE is concerned about the representation of female researchers in the field of computability. In order to increase female participation, the series started the *Women in Computability* (WiC) programme in 2007, first funded by the *Elsevier Foundation*, then taken over by the publisher *Elsevier*. We were proud to continue this programme with its successful annual WiC workshop and a grant programme for junior female researchers in 2013.

The organizers of CiE 2013 would like to acknowledge and thank the following entities for their essential financial support (in alphabetic order): the *Association for Symbolic Logic* (ASL), the *Department of Mathematics and its Applications* and the *Department of Informatics, Systems and Communication*, both of the University of Milano-Bicocca, *Elsevier B.V.*, the *European Association for Computer Science Logic* (EACSL), the *European Association for Theoretical Computer Science* (EATCS), *IOS Press*, *Springer-Verlag* and the *University of Milano-Bicocca*. We would also like to acknowledge the support of our non-financial sponsors, the *Association Computability in Europe* (CiE).

June 11, 2013
Milano

Paola Bonizzoni
Vasco Brattka
Gianluca Della Vedova
Benedikt Löwe

Table of Contents

Program Committee

Gerard Alberts	University of Amsterdam
Luís Antunes	Computer Science Dep., University of Porto
Arnold Beckmann	Swansea University
Laurent Bienvenu	LIAFA
Paola Bonizzoni	universit di Milano-Bicocca
Vasco Brattka	University of Cape Town
Cameron Buckner	UH
Bruno Codenotti	CNR
Stephen Cook	University of Toronto
S. Barry Cooper	University of Leeds
Ann Copestake	University of Cambridge
Erzsébet Csuhaj-Varjú	Computer and Automation Research Institute, Hungarian Academy of Sciences
Anuj Dawar	University of Cambridge
Liesbeth De Mol	Universiteit Gent
Gianluca Della Vedova	Dipartimento di Statistica, Univ. degli Studi Milano-Bicocca
Jérôme Durand-Lose	LIFO - U. D'Orléans
Viv Kendon	School of Physics and Astronomy, University of Leeds
Bjoern Kjos-Hanssen	University of Hawai'i
Antonina Kolokolova	Memorial University of Newfoundland
Benedikt Löwe	Universiteit van Amsterdam
Giancarlo Mauri	University of Milano-Bicocca
Rolf Niedermeier	TU Berlin
Geoffrey K. Pullum	University of Edinburgh
Nicole Schweikardt	Univ. Frankfurt
Sonja Smets	Universiteit Groningen
Susan Stepney	York University
S P Suresh	Chennai Mathematical Institute
Peter Van Emde Boas	ILLC-FNWI-Universiteit van Amsterdam (emeritus)

Additional Reviewers

A
Abu Zaid, Faried
Aizawa, Kenneth
Allo, Patrick
Almeida, Marco
Aman, Bogdan
Anderson, Matthew
Andrews, Paul
Arkhipov, Alex
Arrighi, Pablo
B
Badillo, Liliana
Barmpalias, George
Barr, Katie
Bauer, Andrej
Bauwens, Bruno
Bernardinello, Luca
Besozzi, Daniela
Block, Alexander
Bonfante, Guillaume
Braverman, Mark
Bredereck, Robert
Brenguier, Romain
Bridges, Douglas
Brijder, Robert
Briseid, Eyvind
Bulteau, Laurent
Bès, Alexis
C
Calvert, Wesley
Caravagna, Giulio
Carl, Merlin
Carlucci, Lorenzo
Carton, Olivier
Case, John
Chen, Jiehua
Chivers, Howard
Colombo, Riccardo
Cormode, Graham
Correia, Luis
D
Dahmani, François

Heunen, Chris
Hickinbotham, Simon
Hines, Peter
Hirschfeldt, Denis
Hoyrup, Mathieu
Hutter And Wen Shao, Marcus
Hölzl, Rupert
Hüffner, Falk

I

Inamdar, Tanmay
Ivan, Szabolcs

K

Kabanets, Valentine
Kalimullin, Iskander
Kari, Jarkko
Kari, Lila
Kato, Yuki
Kawamura, Akitoshi
Kent, Tom
Kihara, Takayuki
Kishida, Kohei
Kleijn, Jetty
Klincewicz, Michal
Kolde, Raivo
Komusiewicz, Christian
Kullmann, Oliver

L

Lange, Karen
Lazic, Ranko
Le Gloannec, Bastien
Leal, Raul
Leporati, Alberto
Li, Zhenhao
Loeb, Iris
Lovett, Neil

M

Malcher, Andreas
Manea, Florin
Manzoni, Luca
Margenstern, Maurice
Marks, Andrew
Martiel, Simon
Martini, Simone
Maurino, Andrea
Michaelson, Greg

Miller, Russell
Montalban, Antonio
Moore, Cris
Morozov, Andrei
Mota, Francisco
Mummert, Carl
N
Nguyen, Nam
Nguyen, Paul
Nichterlein, André
Nies, Andre
Nobile, Marco
Noessner, Jan
Nordvall Forsberg, Fredrik
O
O'Keefe, Simon
Okhotin, Alexander
P
Panangaden, Prakash
Paun, Gheorghe
Pavesi, Giulio
Pescini, Dario
Pike, David
Pirola, Yuri
Porreca, Antonio
Poulding, Simon
Pouly, Amaury
Primiero, Giuseppe
Puzarenko, Vadim
R
Rizzi, Raffaella
Romashchenko, Andrei
Rothe, Jrg
Russell, Benjamin
Rute, Jason
S
Salomaa, Kai T.
Sampson, Adam
San Mauro, Luca
Sankur, Ocan
Sapir, Mark
Savani, Rahul
Sciortino, Marinella
Seisenberger, Monika
Seki, Shinnosuke

Sequoiah-Grayson, S
Sergioli, Giuseppe
Seyfferth, Benjamin
Shafer, Paul
Shen, Alexander
Shen, Sasha
Shlapentokh, Alexandra
Simmons, Harold
Simon, Sunil Easaw
Simonnet, Pierre
Skordev, Dimiter
Solomon, Reed
Sorbi, Andrea
Sorge, Manuel
Soskova, Alexandra
Sourabh, Sumit
Souto, André
Sprevak, Mark
Stamatiou, Yannis
Stannett, Mike
Steffen, Bernhard
Stephan, Frank
Suchy, Ondra
Szymanik, Jakub

T

Thierauf, Thomas
Thomson And Rajeev Gore, Jimmy
Tsigaridas, Elias

U

Uckelman, Joel

V

van Bevern, René
van Den Berg, Benno
Vatev, Stefan
Verlan, Sergey
Vicary, Jamie

W

Wareham, Todd
Weihrauch, Klaus
Weller, Mathias

Y

Yakoubsohn, Jean-Claude
Yokoyama, Keita

Z

Zandron, Claudio

Ziegler, Martin
Zoppis, Italo

Efficient Computation of the Gap-Weighted Subsequence Kernel*

Slimane Bellaouar[1], Hadda Cherroun[1], and Djelloul Ziadi[2]

[1] Laboratoire LIM, Université Amar Telidji, Laghouat, Algérie
{s.bellaouar,hadda_cherroun}@mail.lagh-univ.dz
[2] Laboratoire LITIS - EA 4108, Université de Rouen, Rouen, France
Djelloul.Ziadi@univ-rouen.fr

Abstract. In this paper, we present a novel approach to compute efficiently the gap weighted subsequence kernel (GWSK). We have started by the construction of a *match list* $L(s,t) = \{(i,j) : s_i = t_j\}$ where s and t are the strings to be compared. Then, we have constructed a *layered range tree* and a *list of lists*. The whole process takes $O(|L| \log |L| + pK)$ time and $O(|L| \log |L| + K)$ space, where $|L|$ is the size of the match list, p is the length of the GWSK and K is the total reported points by range queries over all the entries of the match list.

Keywords: string kernel, computational geometry, layered range tree, range query

1 Introduction

Kernel methods [1] were proposed as an alternative solution to the limitation of traditional machine learning algorithms applied, solely, on linear separable problems. They project the data into a high dimensional feature space where linear learning machines based on algebra, geometry and statistics can be applied. Hence, non-linear relations can be discovered. Moreover, the kernel methods enables other types of data (biosequences, images, graphs, ...) to be processed.

Strings are considered among the important data types. Therefore, a great effort of research has been devoted to string kernels that are widely used in the fields of bioinformatics and natural language processing. The philosophy of all string kernels can be reduced to different ways to count common substrings or subsequences that occur in the two strings to compare, say s and t.

There are two main approaches to improve the computation of the gap-weighted subsequence kernel (GWSK). The first one is based on dynamic programming paradigm; Lodhi et al. [2] have applied dynamic programming paradigm to the suffix version of the GWSK. Later, Rousu and Shawe-Taylor [3] have proposed an improvement to the dynamic programming approach. They have used a set of match lists combined with a sum range tree. The trie-based approach [4]

* This work is supported by the MESRS - Algeria under Project 8/U03/7015.

is based on depth first traversal on an implicit trie data structure. The number of gaps is restricted, so the computation is approximate.

Motivated by the efficiency of the computation, a key property of the kernel methods, in this paper, we will focus on improving the GWSK computation. We begin by the construction of a *match list* $L(s,t) = \{(i,j) : s_i = t_j\}$ that contains only the required data to the computation. The main idea is that the GWSK computation corresponds to 2-dimensional range queries on a layered range tree (a range tree enhanced by the fractional cascading technique). The final data structure built is a list of lists. The overall time complexity is $O(|L| \log |L| + pK)$, where $|L|$ is the size of the match list and K is the total reported points by range queries over all the entries of the match list.

The rest of this paper is organized as follows. Section 2 deals with some concept definitions used in the other sections. In section 3, we recall formally the GWSK computation. We review three efficient computations of the GWSK, namely, dynamic programming, trie-based and sparse dynamic programming approaches. Section 4 is devoted to the presentation of our contribution. Section 5 includes conclusions and discussion.

2 Preliminaries

This section deals with required concepts to understand the rest of this paper. Let Σ be an alphabet of a finite set of symbols. The number of symbols in Σ is denoted by $|\Sigma|$. $|s|$ denotes the length of the string s. Σ^n denotes the set of all finite strings of length n, and Σ^* denotes the set of all strings. The notation $[s = t]$ is a boolean function that returns

$$\begin{cases} 1 & \text{if } s \text{ and } t \text{ are identical;} \\ 0 & \text{otherwise.} \end{cases}$$

The i^{th} element of the word s is denoted by s_i. The string $s(i:j)$ denotes the substring $s_i s_{i+1} ... s_j$ of s. Accordingly, the string t is a substring of a string s if there are strings u and v such that $s = utv$ (u and v can be empty). The substrings of length n are referred to as n-grams (or n-mers).

The string t is a subsequence of s if there exists an increasing sequence of indices $I = (i_1, ..., i_{|t|})$ in s, $(1 \leq i_1 < ... < i_{|t|} \leq |s|)$ such that $t_j = s_{i_j}$, for $j = 1, ..., |t|$. In the literature, we use $t = s(I)$ if t is a subsequence of s in the positions given by I. The empty string ϵ is indexed by the empty tuple. The length of the subsequence t is denoted by $|t| = |I|$ which is the number of indices, while $l(I) = i_{|t|} - i_1 + 1$ refers to the number of characters of s covered by the subsequence t.

3 Gap-Weighted Subsequence Kernels

The GWSK adopts a new weighting method that reflects the degree of contiguity of a subsequence in the string. In order to measure the distance of non contiguous

elements of the subsequence, a gap penalty $\lambda \in]0,1]$ is introduced. Formally, the mapping function $\phi^p(s)$ in the feature space F can be defined as follows:

$$\phi^p_u(s) = \sum_{I:u=s(I)} \lambda^{l(I)},\ u \in \Sigma^p.$$

The associated kernel can be written as:

$$K_p(s,t) = \langle \phi^p(s), \phi^p(t) \rangle = \sum_{u \in \Sigma^p} \sum_{I:u=s(I)} \sum_{J:u=t(J)} \lambda^{l(I)+l(J)}.$$

A suffix kernel is defined to assist in the computation of the GWSK. The associated embedding is given by:

$$\phi^{p,S}_u(s) = \sum_{I \in I_p^{|s|}:u=s(I)} \lambda^{l(I)}, u \in \Sigma^p,$$

where I_p^k denotes the set of p-tuples of indices I with $i_p = k$ (see Section 2). The associated kernel can be defined as follows:

$$\begin{aligned} K^S_p(s,t) &= \langle \phi^{p,S}(s), \phi^{p,S}(t) \rangle \\ &= \sum_{u \in \Sigma^p} \phi^{p,S}_u(s).\phi^{p,S}_u(t). \end{aligned}$$

The GWSK can be expressed in terms of its suffix version as follows:

$$K_p(s,t) = \sum_{i=1}^{|s|} \sum_{j=1}^{|t|} K^S_p(s(1:i), t(1:j)), \tag{1}$$

with $K^S_1(s,t) = [s_{|s|} = t_{|t|}]\ \lambda^2$.

3.1 Naive Implementation

The computation of the similarity of two strings (sa and tb) is conditioned by their final symbols. In the case where $a = b$, we have to sum kernels of all prefixes of s and t. Hence, a recursion has to be devised:

$$K^S_p(sa, tb) = [a = b] \sum_{i=1}^{|s|} \sum_{j=1}^{|t|} \lambda^{2+|s|-i+|t|-j} K^S_{p-1}(s(1:i), t(1:j)). \tag{2}$$

The computation of equation (2) leads to a complexity of $O(p(|s|^2|t|^2))$.

3.2 Efficient Implementations

We will present three methods that compute the GWSK efficiently. The first is based on a dynamic programming approach [2], the second is the trie-based method [4] and the third is a sparse dynamic programming based approach [3].

Dynamic Programming Approach. The starting point of the dynamic programming approach is the suffix recursion given in equation (2). From this equation, we can consider a separate dynamic programming table DP_p for storing the double sum:

$$DP_p(k,l) = \sum_{i=1}^{k} \sum_{j=1}^{l} \lambda^{k-i+l-j} \, K_{p-1}^{S}(s(1:i), t(1:j)). \tag{3}$$

It is easy to observe that: $K_p^S(sa, tb) = [a = b] \, \lambda^2 \, DP_p(|s|, |t|))$.
Computing ordinary DP_p for each (k, l) would be inefficient. So we can devise a recursive version of equation (3) with a simple counting device:

$$\begin{aligned} DP_p(k,l) = K_{p-1}^{S}(s(1:k), t(1:l)) + \lambda DP_p(k-1,l) + \\ \lambda DP_p(k,l-1) - \lambda^2 DP_p(k-1,l-1). \end{aligned}$$

Consequently, the complexity of the GWSK is $O(p\,|s||t|)$.

Trie-based Approach. This approach is based on search trees known as tries, introduced by E. Fredkin in 1960. The key idea of the trie-based approach is that leaves play the role of the feature space indexed by the set Σ^p. In the literature, there exists a variant of trie-based gap weighted subsequence kernels. For instance the (p, m)-mismatch string kernel [4] and restricted GWSK [5].

In the present section, we try to describe a trie-based GWSK presented in [3] that slightly differ from the one cited above [5]. Given that each node in the trie corresponds to a co-occurrence between strings, the algorithm stores all matches $s(I) = u_1 \cdots u_q$, $I = i_1 \cdots i_q$ in such node. In parallel, it will maintain a list of alive matches $L_s(u, g)$ that records the last index i_q (g is the number of gaps in the match). Notice that in the same list we are able to record many occurrences with different gaps. Similarly, the algorithm is applied to the string t. The process will continue until achieving the depth p where the kernel will be evaluated as follows:

$$K_p(s,t) = \sum_{u \in \Sigma^p} \phi_u^p(s)\phi_u^p(t) = \sum_{u \in \Sigma^p} \sum_{g_s, g_t} \lambda^{g_s+p} |L_s(u, g_s)| \cdot \lambda^{g_t+p} |L_t(u, g_t)|.$$

Given that, there exists $\binom{p+m}{m}$ different entries at leaf nodes, the worst-case time complexity of the algorithm is $O(\binom{p+m}{m} \, (|s| + |t|))$.

Sparse Dynamic Programming Approach. This approach is built on the fact that in many cases, most of the entries of the DP matrix are zero and do not contribute on the result. Rousu and Shawe-Taylor [3] have proposed a solution using two data structures. The first one is a set of match lists instead of the K_p^S matrix. The second one is a range sum tree, which is a B-tree, that replaces the DP_p matrix. It is used to return the sum of n values within an interval in $O(\log n)$ time. Their algorithm runs in $O(p|L| \log \min(|s|, |t|))$, where $L = \{(i, j) | s_i = t_j\}$.

4 List and Layered Range Tree based Approach

Looking forward to improving the complexity of GWSK, our approach is based on two observations. The first one concerns the computation of $K_p^S(s,t)$ that is required only when $s_{|s|} = t_{|t|}$. Hence, we have kept only a list of index pairs of these entries rather than the entire suffix table, $L(s,t) = \{(i,j) : s_i = t_j\}$. If we consider the example which computes $K_p(gatta, cata)$, the list generated is

$$L(gatta, cata) = \{(2,2),(5,2),(3,3),(4,3),(2,4),(5,4)\}.$$

In the rest of the paper, while measuring the complexity of different computations, we will consider, $|L|$, the size of the match list $L(s,t)$ as the parameter indicating the size of the input data.

The complexity of the naive implementation of the list version is $O(p|L|^2)$, and it seems not obvious to compute $K_p^S(s,t)$ efficiently on a list data structure. In order to address this problem, we have made a second observation that the suffix table can be represented as a two-dimensional space (plane) and the entries where $s_{|i|} = t_{|j|}$ as points in this plane.

At the light of this observation, the computation of $K_p^S(s,t)$ can be interpreted as an orthogonal range query. In the literature, there exist several data structures that are used in computational geometry. We have examined a spatial data structure known as Kd-tree [6, 7, 8]. It records a total time cost of $O(p(|L|\sqrt{|L|} + K))$ for computing the GWSK, where K is the total of the reported points. It is clear that this relative amelioration is not sufficiently satisfactory. So we adopted another spatial data structure, called range tree [7, 8, 9, 10], which has better query time for rectangular range queries. We will describe such data structure and its relationship with GWSK in the following sub sections.

4.1 Suffix Table Representation

The entries (k,l) in $L(s,t)$ correspond to a set S of points in the plane, where the index pairs (k,l) play the role of the point coordinates. The set S is represented by a 2-dimensional range tree, where nodes represent points in the plane. Thereby, representing the suffix table tend to be the construction of a 2-dimensional range tree. A range tree, denoted by $\mathcal{RT}$ is primarily a balanced binary search tree (BBST) augmented with an associated data structure. In order to build such data structure, first, we consider the set S_x of the first coordinate (x-coordinate) values of all the points in S. Thereafter, a BBST called x-$\mathcal{RT}$ is constructed with points of S_x in the leaves. Both internal or leaf nodes v of x-$\mathcal{RT}$ are augmented by a 1-dimensional range tree, it can be a BBST or a sorted array, of a canonical subset $P(v)$ on y-coordinates, denoted by y-$\mathcal{RT}$. The subset $P(v)$ is the points stored in the leaves of the sub tree rooted at the node v. Figure 1 illustrates the construction process of a 2-dimensional range tree.

In the case where two points have the same x or y-coordinate, we have to define a total order by using a lexicographic one. It consists to replace the real

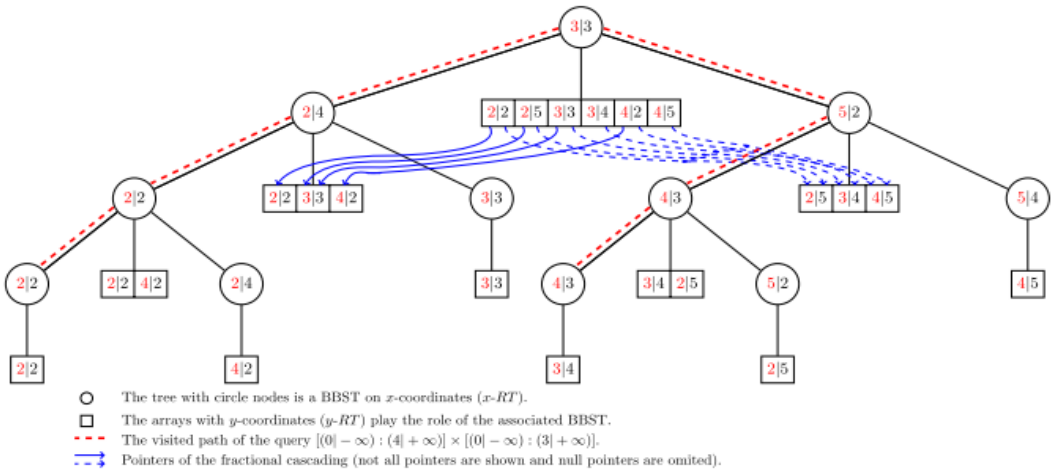


Fig. 1. Layered range tree $\mathcal{RT}$ related to K_p^S(gatta, cata)

number by a composite-number space [7]. The composite number of two reals x and y is denoted by $x|y$, so for two points, we have:

$$(x|y) < (x'|y') \Leftrightarrow x < x' \vee (x = x' \wedge y < y').$$

In such situation, we have to transform the range query $[x_1 : x_2] \times [y_1 : y_2]$ related to a set of points in the plane to the range query $[(x_1| - \infty) : (x_2| + \infty)] \times [(y_1| - \infty) : (y_2| + \infty)]$ related to the composite space.

Based on the algorithm analysis of computational geometry algorithms, our 2-dimensional range tree requires $O(|L| \log |L|)$ storage and can be constructed in $O(|L| \log |L|)$ time. This leads to the following lemma.

Lemma 1. *Let s and t be two strings and $L(s,t) = \{(i,j) : s_i = t_j\}$ the match list associated to the suffix version of the GWSK. A range tree for $L(s,t)$ requires $O(|L| \log |L|)$ storage and takes $O(|L| \log |L|)$ construction time.*

4.2 Location of Points in a Range

We recall that computing the recursion for the GWSK given by the equation (2) can be interpreted as the evaluation of a 2-dimensional range query applied to a 2-dimensional range tree. Such evaluation locates all points that lie in the specified range.

A useful idea, in terms of efficiency, consists on treating a rectangular range query as a two nested 1-dimensional queries. In other words, let $[x_1 : x_2] \times [y_1 : y_2]$ be a 2-dimensional range query, we first ask for the points with x-coordinates in the given 1-dimensional range query $[x_1 : x_2]$. Consequently, we select a collection of $O(\log |L|)$ subtrees. We consider only the canonical subset of the resulted

subtrees, which contains, exactly, the points that lies in the x-range $[x_1 : x_2]$. At the next step, we will only consider the points that fall in the y-range $[y_1 : y_2]$. The total task of a range query can be performed in $O(\log^2 |L| + k)$ time, where k is the number of points that are in the range. We can improve it by enhancing the 2-dimensional range tree with the fractional cascading technique which is described in the following subsection.

4.3 Fractional Cascading

The key observation made during the invocation of a rectangular range query is that we have to search the same range $[y_1 : y_2]$ in the associated structures y-$\mathcal{RT}$ of $O(\log |L|)$ nodes found while querying the x-$\mathcal{RT}$ by the range query $[x_1 : x_2]$. Moreover, there exists an inclusion relationship between these associated structures. The goal of the fractional cascading consists on executing the binary search only once and use the result to speed up other searches without expanding the storage by more than a constant factor.

The application of the fractional cascading technique introduced by [11] on a range tree creates a new data structure so called layered range tree. We will illustrate such technique through a simple of GWSK computing in Fig. 1.

Using this technique, the rectangular search query time becomes $O(\log(|L| + k)$, where k is the number of reported points. For the computation of $K_p^S(s,t)$ we have to consider $|L|$ entries of the match list. The process iterates p times, therefore, we get a time complexity of $O(p|L| \log |L| + K)$ for evaluating the GWSK, where K is the total of reported points over all the entries of $L(s,t)$. This result combined to that of Lemma. 1 lead to the following Lemma:

Lemma 2. *Let s and t be two strings and $L(s,t) = \{(i,j) : s_i = t_j\}$ the match list associated to the suffix version of the GWSK. A layered range tree for $L(s,t)$ uses $O(|L| \log |L|)$ storage and it can be constructed in $O(|L| \log |L|)$ time. With this layered range tree, the GWSK of length p can be computed in $O(p(|L| \log |L| + K))$, where K is the total number of reported points over all the entries of $L(s,t)$.*

4.4 List of lists Building and GWSK Computation

Another observation leads us to pursue our line of reasoning about the improvement of the GWSK computation complexity. It is obvious to state that point coordinates, in our case, in the plane remain unchanged during the entire processing. So instead of invoking the 2-dimensional range query multiple times according to the evolution of the parameter p, it is more beneficial if we do the computation only once. Accordingly, in this phase, we extend our match list to be a list of lists (Fig. 2), where each entry (k,l) points to a list that contains all the points that lie in the corresponding range. Algorithm 1 builds this list of lists. The complexity of the construction of the list of lists is the complexity of invoking the 2-dimensional range query over all the entries of the match list. This leads to $O(|L| \log |L| + K)$ time complexity.

Algorithm 1 List of Lists Creation

Require: match list $L(s,t)$ and Layered Range Tree $\mathcal{RT}$
  for each entry $(k,l) \in L(s,t)$ **do**
    {Preparing the range query}
    $x_1 \leftarrow 0$
    $y_1 \leftarrow 0$
    $x_2 \leftarrow k-1$
    $y_2 \leftarrow l-1$
    relatedpoints $\leftarrow$ 2D-RANGE-QUERY($\mathcal{RT}$, $[(x_1|-\infty):(x_2|+\infty)] \times [(y_1|-\infty):(y_2|+\infty)]$
    while There exists $(i,j) \in$ relatedpoints **do**
      add (i,j) to (k,l)-list
    end while
  end for
Ensure: List of Lists $LL(s,t)$: The match list augmented with lists containing reported points

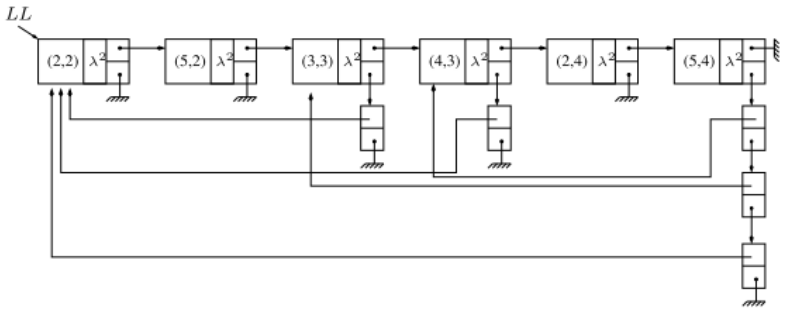


Fig. 2. List of lists inherent to K_1^S(gatta,cata).

Once the list of lists constructed, the GWSK computation will sum over all the reported points stored on it. The process is described in Algorithm 2. The cost of this computation is $O(K)$. Since we will evaluate the GWSK for $p \in [1..\min(|s|,|t|)]$, this leads to a complexity of $O(pK)$. So the over all complexity is $O(|L|\log|L| + pK)$ which include the construction of the list of lists and the computation of GWSK in the strict sense. This leads to the following theorem that summarizes the result for the computation of the GWSK.

Theorem 1. *Let s and t be two strings and $L(s,t) = \{(i,j) : s_i = t_j\}$ the match list associated to the suffix version of the GWSK. A layered range tree and a list of lists for $L(s,t)$ require $O(|L|\log|L| + K)$ storage and they can be constructed in $O(|L|\log|L| + K)$ time. With these data structures, the GWSK of length p*

can be computed in $O(|L| \log |L| + pK)$, where K is the total number of reported points over all the entries of $L(s,t)$.

Algorithm 2 GWSK computation

Require: List of Lists $LL(s,t)$, subsequence length p and penalty coefficient λ
  for q=1:p **do**
    {Initialization}
    $K(q) \leftarrow 0$
    $KPS(1:|max|) \leftarrow 0$
    for each entry $(k,l) \in LL(s,t)$ **do**
      for each entry $r \in (k,l) - list$ **do**
        $(i,j) \leftarrow r.Key$
        $KPS_{(i,j)} \leftarrow r.Value$
        $KPS(k,l) \leftarrow KPS(k,l) + \lambda^{k-i+l-j}\ KPS_{(i,j)}$
      end for
      $K(q) \leftarrow K(q) + KPS(k,l))$
    end for
    {Preparing $LL(s,t)$ For the next computation}
    for each entry $(k,l) \in KPS$ **do**
      Update $LL(k,l)$ with $KPS(k,l)$
    end for
  end for
Ensure: Kernel values $K_q(s,t) = K(q) : q = 1, \ldots, p$

5 Conclusion

We presented a novel algorithm that efficiently computes the gap weighted subsequence kernel (GWSK). Our approach is refined over three phases. We begin by the construction of a match list $L(s,t)$ that contains, only, the information that contributes in the result. In order to locate, efficiently, the related positions for each entry of the match list, we have constructed a layered range tree. At last, we have built a list of lists to compute efficiently the GWSK. The Whole task takes $O(|L| \log |L| + pK)$ time and $O(|L| \log |L| + K)$ space, where p is the length of the GWSK and K is the total number of reported points.

The reached result gives evidence of an asymptotic complexity improvement compared to that of a naive implementation of the list version $O(p\,|L|^2)$. On the other hand, our intermediate data structure, the layered range tree is output sensitive. It means that our computation implies, only, required data. However, this dictates to conduct empirical analysis to compare our contribution to other approaches. This will be the subject of a future research.

Nevertheless, based on the asymptotic complexities of the different approaches and the experiments presented in [3], we make some discussions. The dynamic programming approach is faster when the DP_p table is nearly full. This case is achieved on short strings and on long strings if the alphabet is small. The

trie-based approach is faster on medium-sized alphabets but it suffers from gap length restriction. Furthermore, recall that our approach and the sparse dynamic programming one are proposed in the context where the most of the entries of the DP_p table are zero. This case occurred for large-sized alphabets. From the asymptotic complexity of the sparse approach, $O(p|L| \log \min(|s|, |t|))$, it is clear that its efficiency depends on the size of the strings. For our approach it depends only on the number of common subsequences. Under these conditions our approach outperforms for long strings. Theses discussions will be validated by a future empirical study.

A noteworthy advantage is that our approach separates the process of required data location from the strict computation one. This separation limits the impact of the length of the GWSK on the computation. It have influence, only, on the strict computation process. Moreover, such separation property can be favorable if we assume that the problem is multi-dimensional, e.g. comparing several strings in the same time. In terms of complexity, this can have influence, only, on the location process by a logarithmic factor. Indeed, the layered range tree can report points that lies in a rectangular range query in $O(\log^{d-1} |L| + k)$, in a d-dimensional space. At the implementation level, great programming effort is supported by well-studied and ready to use computational geometry algorithms. Hence, the emphasis is shifted to a variant of string kernel computations that can be easily adapted.

References

[1] Cristianini, N., Shawe-Taylor, J.: An introduction to support Vector Machines: and other kernel-based learning methods. Cambridge University Press, New York, NY, USA (2000)

[2] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. J. Mach. Learn. Res. **2** (March 2002) 419–444

[3] Rousu, J., Shawe-Taylor, J.: Efficient computation of gapped substring kernels on large alphabets. J. Mach. Learn. Res. **6** (December 2005) 1323–1344

[4] Leslie, C., Eskin, E., Noble, W.: Mismatch String Kernels for SVM Protein Classification. In: Neural Information Processing Systems 15. (2003) 1441–1448

[5] Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA (2004)

[6] Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9) (September 1975) 509–517

[7] Berg, M.d., Cheong, O., Kreveld, M.v., Overmars, M.: Computational Geometry: Algorithms and Applications. 3rd ed. edn. Springer-Verlag TELOS, Santa Clara, CA, USA (2008)

[8] Samet, H.: The design and analysis of spatial data structures. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1990)

[9] Bentley, J.L.: Decomposable searching problems. Inf. Process. Lett. **8**(5) (1979) 244–251

[10] Bentley, J.L., Maurer, H.A.: Efficient Worst-Case Data Structures for Range Searching. Acta Informatica **13** (1980) 155–168

[11] Chazelle, B., Guibas, L.J.: Fractional cascading: I. a data structuring technique. Algorithmica **1**(2) (1986) 133–162

Reaction systems with constrained environment

Paolo Bottoni[1] and Anna Labella[1]

Università di Roma "Sapienza" (Italy) (bottoni,labella)@di.uniroma1.it

Abstract. Reaction systems model living organisms as systems sustaining themselves through reaction processes and receiving contributions from the environment. In the classical model, the system is completely open and the environment is completely unpredictable, contributing to system evolution via a set-theoretic union operation. We discuss variants in which the environment can be modeled as a function or relation involving the state of the system and in which communication occurs through specific channels. Moreover, we discuss models of environment contribution via intersection and symmetrical difference, rather than union.

1 Introduction

Reaction systems [6], a novel paradigm of nature-inspired computing, model living organisms by defining self-sustainable sets of reactions which can take place inside of them. In particular, a reaction can occur if the system's current configuration: (1) contains all the reactants needed for it, and (2) does not contain any inhibitor preventing it. As a result, the original configuration is replaced with the products of the reaction. Hence, a reaction does not describe a mechanism of consumption and production of individual, distinct resources. Rather, reaction systems can be considered as defining a type calculus in which some types of element are produced only if some types of element, the *reactants*, are available and other types, the *inhibitors*, are absent. The main distinction with respect to traditional resource-based approaches, e.g. multiset rewriting [1, 3, 4, 2, 14], is that no counting of resources occurs, so that all possible reactions in a configuration occur simultaneously without conflicts even if their sets of reactants are not disjoint, and that unused resources are not maintained between reactions: in a reaction system, if a type of element is not produced by one of the reactions occurring on a configuration, it will not appear in the next configuration. Although counting is not required in models based on DNA computing [15], as all necessary copies are assumed to be present, these models still assume that resources persist even if not reproduced, as a system boundaries preserve its identity; a reaction system exists as long as it is able to sustain a set of reactions.

Moreover, organisms are modeled as open systems, accepting any contribution from the environment. Hence. in addition to the reaction mechanism, the evolution of a reaction system depends on the injection of other types of elements from the environment, determining the configuration on which the next set of reactions will occur. In the classical model of reaction systems the environment

is completely unpredictable, i.e. there is no relation between the current system configuration and the contribution provided by the environment. However, it is often the case that the latter presents some form of dependence on the current state of the system. For example in osmotic processes, the concentration of substances at the system boundaries determines the gradient of intake for those substances. In social systems, adversarial or collaborative interactions may develop by having other systems sending input towards the system under study to orient its evolution. Moreover, in classical reaction systems the system is completely open, i.e. whatever the environment contributes becomes part of the system's configuration. Again, systems might present filtering mechanisms which treat input from the environment in a different way, for example as a form of protection, or combine it only with matching or complementary resources.

These considerations motivate us to explore variants which constrain environmental contribution in different ways, without modifying the notion of reaction. First, we consider a model in which the environment contribution depends, either in a deterministic or a non-deterministic way, on the current configuration. We observe that compliance or not with two simple boundary conditions modifies the computational power of the model. Second, we consider that only certain types of elements can be contributed by the environment, whereas the others have to be produced by the system itself, and show that this restriction does not extend the computational power of functional environments. Third, we study different ways in which the contribution of the environment can be integrated in the configuration. While the classical model is based on set-theoretic union, we explore mechanisms based on intersection and symmetrical difference, discussing the possibility of forcing the system to perform only some types of evolution.

Paper organisation. After recalling the classical model of reaction systems in Section 2, we explore three types of restrictions on the environment in Section 3, and the use of different set-theoretical operations in Section 4. We discuss related work in Section 5, and conclude the paper and discuss future work in Section 6.

2 Background

We recall here the basic formal notions on reaction systems (see e.g. [6]).

Let S be a finite set of elements, called a *support*. A *reaction in* S is a triple $a = (R_a, I_a, P_a)$, with: $R_a \cup I_a \cup P_a \subseteq S$, $R_a \cap I_a = \emptyset$, and none of R_a, I_a, P_a empty[1]. A reaction a is *enabled* in a configuration[2] $D \subset S$ *iff* $R_a \subseteq D$ *and* $I_a \cap D = \emptyset$. We call R_a the set of *reactants*, I_a the set of *inhibitors* and P_a the set of *products* for the reaction a. We consider sets of reactions, denoted by $A = \{a_1, \ldots, a_n\}$, where each a_i is a reaction in S. Note that any set of reactions can be assumed to act on the same support S, by just taking $S = \bigcup_{a_i \in A} R_{a_i} \cup I_{a_i} \cup P_{a_i}$. Given a configuration D, we call $en_A(D)$ the set of reactions from A enabled in D.

A reaction system is a pair $\mathcal{A} = (S, A)$. A *reaction step* on $D \subset S$ is defined as $D \Rightarrow D' = res_{\mathcal{A}}(D)$, where $res_{\mathcal{A}}(D) = \bigcup_{a \in en_A(D)} P_a$. An *evolution step* is

[1] Having a rule $(X, Y, \emptyset)$ would be the same as lacking rules with $R_a = X$, $I_a = Y$.
[2] D must be a proper non-empty subset of S for any reaction to be enabled in it.

defined as $D' \Rightarrow W = (D' \cup C)$, where C represents the *contribution* from the *environment*. By representing the system overall evolution by a sequence of sets, the composition of a reaction and an evolution step give rise to a subsequence $D \cdot W$. In principle, one could consider the distinct sequences $\mathcal{D}$, $\mathcal{C}$ and $\mathcal{W}$ of reaction steps, contributions, and evolution steps for a system $\mathcal{A}$, respectively. We indicate with $[\alpha]$ the constant repetition of a subsequence α and call *non-terminating* a sequence which does not have the form $\alpha \cdot [\emptyset]$.

We denote by $CISTS(\mathcal{A})$ and $STS(\mathcal{A})$ the sets of sequences of evolution steps (i.e. of all possible $\mathcal{W}$ for $\mathcal{A}$) where $C = \emptyset$ at all steps (i.e. the system is *context-independent*) and with arbitrary C at all steps, respectively. We define $STS(S)$ ($CISTS(S)$) as the set of possible sets of sequences of evolution steps (with $C = \emptyset$ at all steps) for any possible reaction system on S. Note that even if the system produces an empty configuration in a reaction step, the environment can sustain it by injecting new entities.

3 Constraints on the environment

We discuss now different ways in which the contribution of the environment can be constrained. In particular, we consider functional, relational or channelled environments, showing the relevance of boundary conditions in determining the overall sequence of system evolutions. In all these cases, we consider that the contribution of the environment is merged with the product of a reaction system in the classical way, i.e. via the union operation.

3.1 Functional environments

We model the environment contribution as a function $\psi : \wp(S) \to \wp(S)$ on the result of a reaction, where $\wp(S)$ denotes the powerset of S, i.e. the set of all its subsets; hence we have $D' = res_{\mathcal{A}}(W)$, $C = \psi(D')$ and $W' = D' \cup C$. If ψ is such that $\psi(S) = \psi(\emptyset) = \emptyset$, we say that ψ *respects the boundary conditions* (ψ is BC). Conversely, ψ is *non-BC* if $\psi(W) \neq \emptyset$ for at least one of $W = \emptyset, S$.

We now consider a few examples of reaction systems on a set S of two elements, the minimum cardinality of S for which a reaction system can be defined. Any set of reactions A on S has rules of the form (X, Y, Z), with $X \cup Y = S$ and $Z \subseteq S$. We obtain the following possible schemes of reaction systems, where Z_1 and Z_2 are non-empty subsets of S.

1. $A = \{(X, Y, Z_1)\}$.
2. $A = \{(X, Y, Z_1), (Y, X, Z_2)\}$, with $Z_1 \neq Z_2$.
3. $A = \{(X, Y, Z_1), (Y, X, Z_1)\}$.

Example 1. Let ψ be a constant BC function with $\psi(W) = \{z\}$ for $W \subset S$, $W \neq \emptyset, S$, $z \in S$, the cases $z = \emptyset$ or $z = S$ being trivial. Then we have the following $\mathcal{W}$ sequences:

1. For A of type 1, supposing the initial configuration is X, we have

- For $X = \{z\} = Z_1$, $STS(\mathcal{A}) = [X]$.
- For $Z_1 \neq \{z\}$, $STS(\mathcal{A}) = X \cdot S \cdot [\emptyset]$.

2. For A of type 2, we have:
   - For $X = \{z\} = Z_1$, $STS(\mathcal{A}) = [X]$.
   - For $X = \{z\}$, $Y \cap Z_1 \neq \emptyset$, $STS(\mathcal{A}) = X \cdot S \cdot [\emptyset]$.
   - For $X \neq \{z\}$, $X \cap Z_1 = \emptyset$, $STS(\mathcal{A}) = X \cdot Y \cdot Z_2 \cdot \alpha$. The form of α depends on whether Z_2 is a singleton or not.

   Analogous results are obtained by exchanging X with Y and Z_1 with Z_2.
3. For A of type 3, we have:
   - For $X = \{z\}$, $Y \cap Z_1 = \emptyset$, $STS(\mathcal{A}) = [X]$.
   - For $X = \{z\}$, $Y \cap Z_1 \neq \emptyset$, $STS(\mathcal{A}) = X \cdot S \cdot [\emptyset]$.
   - For $X \neq \{z\}$, $X \not\subset Z_1$, $X = Z_2$, $STS(\mathcal{A}) = [X \cdot Y]$.

Example 2. Let ψ be a constant *non-BC* function, so that $\psi(W) = \{z\}$ for $W \subseteq S$, $z \in S$ Then we have the following $\mathcal{W}$ sequences:

1. For A of type 1, supposing the initial configuration is X, we have:
   - For $X = \{z\} = Z_1$, $STS(\mathcal{A}) = [X]$.
   - For $X = \{z\}$, $Y \cap Z_1 \neq \emptyset$, $STS(\mathcal{A}) = [X \cdot S]$.
2. For A of type 2, we have:
   - For $X \neq \{z\}$, $X \not\subset Z_1$, $STS(\mathcal{A}) = X \cdot S \cdot [\{z\}]$.

   Analogous results are obtained by exchanging X with Y and Z_1 with Z_2.
3. For A of type 3, we have:
   - For $X = \{z\}$, $Y \cap Z_1 = \emptyset$, $STS(\mathcal{A}) = [X]$.
   - For $X = \{z\}$, $Y \cap Z_1 \neq \emptyset$, $STS(\mathcal{A}) = [X \cdot S]$.
   - For $X \neq \{z\}$, $X \not\subset Z_1$, $X = Z_2$ $STS(\mathcal{A}) = [X \cdot Y \cdot S]$.

Example 3. Let ψ be the complement function, i.e. $\psi(W) = S \setminus W$ for $W \subseteq S$, which is obviously *non-BC* and models an *osmosis* process. For any reaction system $\mathcal{A}$, starting with X we would have $STS(\mathcal{A}) = X \cdot [S]$.

We call $FSTS^{\psi}(\mathcal{A})$ the set of sequences for a reaction system $\mathcal{A}$ and a functional environment ψ, $FSTS(\mathcal{A})$ the set of sequences for $\mathcal{A}$ and any functional, *non-BC*, environment, and $FSTS(S)$ the set of sequences for any reaction system on S and any functional *non-BC*, environment. Theorems 1 and 2 formalise the relevance of boundary conditions.

Theorem 1. *Given a reaction system $\mathcal{A}$ with functional, BC, environment ψ, there exists a reaction system $\mathcal{A}'$ such that the set of the possible sequences of context-independent evolution steps for $\mathcal{A}'$ is equal to the set of possible sequences of evolution steps for $\mathcal{A}$ and ψ.*

Proof. We build the reaction system $\mathcal{A}' = (S, A')$ such that A' contains a rule for each possible evolution of a configuration, except those which would result in an empty configuration as the result of a reaction step. Since ψ is BC, in this case also the contribution from the environment would be empty. Formally, we write: $\forall W \subset S[(W \neq S \wedge W \neq \emptyset \wedge res_{\mathcal{A}}(W) \neq \emptyset) \implies (W, S \setminus W, res_{\mathcal{A}}(W) \cup \psi(res_{\mathcal{A}}(W))) \in A']$. Note that this is a reaction system, as none of W, $S \setminus W$, $res_{\mathcal{A}}(W) \cup \psi(res_{\mathcal{A}}(W))$ under the assumptions above can be empty. It is immediate to see that $CISTS(\mathcal{A}') = STS(\mathcal{A})$.

Theorem 2. $CISTS(S) \subset FSTS(S) \subset STS(S)$.

Proof. The first inclusion derives from the fact that a context-independent environment can be simulated by a functional *non-BC* environment equipped with the identity function ψ, i.e. $\psi(W) = W$ for $W \subseteq S$. For strictness, note that no sequence in $CISTS(S)$ can contain a subsequence $Y \cdot X$, with $Y = \emptyset$ or $Y = S$, for any $X \subset (S)$, $X \neq \emptyset$, which is instead possible for a system with functional environment (e.g. for a constant function ψ). That $FSTS(S)$ is a proper subset of $STS(S)$ derives from the fact that, while all sequences in $FSTS(S)$ are of course within $STS(S)$, the latter, due to the arbitrary nature of the contribution from an unconstrained environment, contains also sequences with subsequences of the form $S \cdot X \cdot \alpha \cdot S \cdot Y$, with $X, Y \subset S$, $X \neq Y$ which are not possible for a sequence in $FSTS(S)$.

3.2 Relational environments

We model the environment contribution via a relation $\rho \subset \wp(S) \times \wp(\wp(S))$, where the first element of each pair represents the result of a reaction, and the second element is a subset of entities compatible with the current configuration. We then have $D' = res_{\mathcal{A}}(W)$, $C \in \rho(D')$ and $W' = D' \cup C$.

Obviously a functional environment is a particular case of relational environment, where all the second components of ρ are singletons. Moreover, in the case where each second component is the whole powerset $\wp(S)$, we have the classical model of reaction systems, while if each second component is the empty set we have context-independent reaction systems. Interesting effects might be reached by relations that ensure that some inhibitor for some reaction a is always present in each second component, which makes it possible to restrict sequences, so that a never occurs. The dual case where the set of reactants for some reaction a is always provided (and no inhibitor is ever provided) does not necessarily force a to occur, as some inhibitor could be produced by the reaction step anyway.

3.3 Channels

Channels model a situation in which the environment can contribute only some types of elements, and not any possible subset of S. An *input channel* is a subset $\chi \subset S$ such that the environment can contribute only subsets of χ. We then have $D' = res_{\mathcal{A}}(W)$, $C \subset \chi$ and $W' = D' \cup C$. The cases in which the environment is functional or relational result accordingly, i.e. $C = \psi(D') \cap \chi$ or $C = C' \cap \chi$ for some $C' \in \rho(D')$. It is easy to note that channels can be simulated by relations, by defining ρ so that for each $W \subseteq S$ and for each $\chi_i \subset \chi$, we have $(W, \chi_i) \in \rho$.

In a different perspective, one can define *output channels*, so that only some types of elements can be communicated from the reaction system to the environment, for it to evaluate its contribution. In this case, let $\xi \subset S$ be the output channel. The function ψ and the relation ρ will be computed on $D' \cap \xi$. Again, output channels can be directly simulated within the function or the relation.

4 Different forms of contribution

In Section 3, we have considered the classical way of defining the environment contribution via a union operation. In this case, the contribution is always a positive one, even if it can adversely affect the behaviour of the system, for example by introducing inhibitors. We now consider other types of operation, which can intervene to remove from the current configuration some element. We consider in all cases a functional environment.

4.1 Intersection

An intersecting environment models a *synchronisation* process, where only elements which are consistently produced by both the system and the environment are maintained. Again, we model the environment via a function $\psi : \wp(S) \to \wp(S)$ on the result of a reaction, but now we model the evolution process by defining $D' = res_{\mathcal{A}}(W)$ and $W' = D' \cap \psi(D')$. We call $FSTS_{\mathcal{I}}(\mathcal{A})$ the set of all possible sequences of evolutions for a reaction system $\mathcal{A}$ with functional environment and contribution restricted to intersection.

Let us suppose that $CISTS(\mathcal{A})$ does not contain terminating sequences. Then, all its sequences are of the form $S_1 = Z \cdot \alpha \cdot [X]$ (i.e. $res_{\mathcal{A}}(X) = X$) or $S_1 = Z \cdot \alpha \cdot [X \cdot \delta_1 \cdot \cdots \cdot \delta_n]$ (i.e. $res_{\mathcal{A}}(X) = \delta_1$ and $res_{\mathcal{A}}(\delta_n) = X$) for some X with $X \neq \emptyset$ *and* $X \neq S$. For any $X \subseteq S$, such that $X \cap \psi(X) = \emptyset$, any initial sequence of the form $\alpha \cdot X$ can only proceed, when using intersection, as $\alpha \cdot X \cdot [\emptyset]$. We therefore consider the following cases, for $X' = \psi(X)$ and $Y = (X' \cap X) \neq \emptyset$.

1. $Y = X'$. Then the system will produce sequences of the form $S'_1 = Z \cdot \beta \cdot X' \cdot Y \cdot \gamma$, for some sequence γ, if there exists an m such that $\beta = \beta_1 \cdot \cdots \cdot \beta_m$ with $\beta_1 = res_{\mathcal{A}}(Z) \cap \psi(res_{\mathcal{A}}(Z))$ and $res_{\mathcal{A}}(\beta_m) \cap \psi(res_{\mathcal{A}}(\beta_m)) = X$.
2. $Y = X$. Then the system will produce sequences of the form $S'_1 = Z \cdot \beta \cdot [X]$ and sequences of the form form $S'_2 = Z \cdot \beta \cdot X \cdot (\delta_1 \cap \psi(\delta_1)) \cdot \delta'$, for some δ', under the same conditions for β.

For the case $(Y \neq X)$ *and* $(Y \neq X')$, no particular conclusion can be drawn. We can now prove Theorem 3.

Theorem 3. *The following hold:*

1. *For any $X, X' \subset S$ with $\psi(X) = X'$, $FSTS_{\mathcal{I}}(\mathcal{A})$ admits a sequence of the form $\alpha \cdot [X']$* iff *$CISTS(\mathcal{A})$ admits a sequence of the form $[X \cdot \beta \cdot X']$.*
2. *If all the sequences in $CISTS(\mathcal{A})$ are of the form $\alpha \cdot [X \cdot \beta]$ and $\psi(X) \supset X$ for all such X, then $FSTS_{\mathcal{I}}(\mathcal{A})$ admits non terminating sequences.*

One can study the relation between contribution via union and intersection with respect to different characteristics of the reaction systems. A duality result ensues for a particular case. For a set $X \subset S$, we denote by $\overline{X}$ its complement with respect to S. We say that a reaction system $\mathcal{A} = (S, A)$ is *autodual* if it enjoys the following property: $a = (R_a, I_a, P_a) \in A \Leftrightarrow a' = (I_a, R_a, \overline{P_a}) \in A$.

Proposition 1 establishes a connection between the autoduality property of reaction systems and the set-theoretic duality of intersection and union. We denote by $FSTS^{\psi}(\mathcal{A}, W)$ the set of evolution sequences of a reaction system $\mathcal{A}$, starting from an initial configuration W and with a functional environment ψ contributing via union. Similarly, we use $FSTS^{\psi}_{\mathcal{I}}(\mathcal{A}, W)$ for the case of a functional environment contributing via intersection.

Proposition 1. *Let $\mathcal{A} = (S, A)$ be an autodual reaction system, D a configuration of it and ψ a functional environment. Then there exists a functional environment ψ' such that $D \cdot \delta_1 \cdots \cdots \delta_n \in FSTS^{\psi}(\mathcal{A}, D) \Leftrightarrow \overline{D} \cdot \overline{\delta_1} \cdots \cdots \overline{\delta_n} \in FSTS^{\psi'}_{\mathcal{I}}(\mathcal{A}, \overline{D})$.*

Proof. We define ψ' as follows: for each $W \subset S$ we have $\psi'(W) = \overline{\psi(\overline{W})}$. Then given a configuration D, for each $a \in en_A(D)$ we have: $D \cap I_a = \emptyset$, from which we infer $\overline{D} \supseteq I_a = R_{a'}$. Hence, an evolution step applying exactly a in the configuration D and using ψ via union will produce $W = P_a \cup \psi(P_a)$, while in the configuration $\overline{D}$, using ψ' and intersection, we will have $W' = \overline{P_a} \cap \overline{\psi(\overline{\overline{P_a}})} = \overline{P_a} \cap \overline{\psi(P_a)} = \overline{W}$ for De Morgan's law. The thesis follows by extension to the union of the products of all rules in $en_A(D)$.

4.2 Symmetrical difference

An environment which contributes via symmetric difference between the result of a process and the result of the function models a *challenging* process, where an element can remain in the system only if it is produced by an identifiable source, the system itself or the environment. We define the operation of symmetrical difference, denoted by Δ, as $X \Delta Y \equiv X \setminus Y \cup Y \setminus X$. As before, we model the environment through a function $\psi : \wp(S) \to \wp(S)$ but now the evolution process is modeled as $D' = res_{\mathcal{A}}(W)$, $W' = D' \Delta \psi(D')$. We call $FSTS_{\Delta}(\mathcal{A})$ the set of all possible sequences of evolutions for a reaction system $\mathcal{A}$ with functional environment and contribution restricted to symmetrical difference.

Note that an osmotic process, modeled in Section 3.1 by the function calculating the complement, will produce the same sequences if the contribution of the environment is produced via the union or via the symmetrical difference operator. In particular, if ψ calculates the complement, except for the cases defined by the boundary conditions, then all sequences have the form $Z \cdot [\emptyset]$. We are therefore interested in the case where $\psi(X) \neq S \setminus X$ for all $X \subset S$.

Under this assumption, we study the differences between the three types of contribution. As discussed in Section 3.2 for the case of relations, an environment contributing with union can add elements to a configuration of a system, so that a functional environment can systematically forbid a given reaction a in a configuration by always introducing at least one inhibitor, as well as ensuring that the set of reactants for a is always present, but it cannot allow such a reaction if at least one inhibitor for a is already present in the configuration, as the result of a reaction step. On the other hand, an environment contributing with intersection can exclude some elements from a configuration of a system, so that it can allow a given reaction a in a configuration containing all the

necessary reactants by removing any inhibitor already present, and it can forbid an a by removing some of its reactants, i.e. not presenting it for contribution. However, neither union nor intersection can be used to force the system to always apply a certain reaction, i.e. to define a self-sustaining behaviour in presence of contributions from the environment.

On the contrary, an environment contributing with symmetrical difference can remove inhibitors and introduce reactants for a at the same time, thus forcing the execution of a on any configuration. Let $a = (R_a, I_a, P_a)$ be the reaction we want to be forcefully executed. Then, for each $W \subset S$, we define ψ so that $\psi(W) = (R_a \setminus W) \cup (I_a \cap W)$, i.e. ψ provides all the missing reactants and all and only the inhibitors already present in W. The operator Δ will then take care of removing all the inhibitors and add all the needed reactants.

Moreover, the evaluation of the contribution through symmetrical difference can emulate both the cases of contribution through union or intersection.

In the first case, let ψ be the function defining an environment which contributes through union. We define a new function ψ' such that for each $W \subset S$, $\psi'(W) = \psi(W) \setminus W$. In this case, for each evolution step in $FSTS^{\psi}(\mathcal{A})$ there is an identical evolution step in $FSTS^{\psi'}_{\Delta}(\mathcal{A})$. In a similar way, let ψ be the function defining an environment which contributes via intersection. We define a new function ψ' such that for each $W \subset S$, $\psi'(W) = W \setminus \psi(W)$. In this case, for each evolution step in $FSTS^{\psi}_{\mathcal{I}}(\mathcal{A})$ there is an identical evolution step in $FSTS^{\psi'}_{\Delta}(\mathcal{A})$.

Theorem 4 follows from the arguments above.

Theorem 4. *Given a reaction system $\mathcal{A}$, such that no two rules $a_i, a_j \in A$ have $R_{a_i} \subseteq R_{a_j}$ and $I_{a_i} = I_{a_j}$, we have $FSTS(\mathcal{A}) \subseteq FSTS_{\Delta}(\mathcal{A})$ and $FSTS_{\mathcal{I}}(\mathcal{A}) \subseteq FSTS_{\Delta}(\mathcal{A})$. If $\mathcal{A}$ admits evolution steps where more than one rule can occur, both inclusions are strict.*

Proof. We only need to prove strictness of the inclusion. This derives by extending the ability of symmetrical difference to force the execution of a rule to also prevent the execution of any other rule. Suppose that $A = \{a_1, \ldots, a_n\}$, that we want to force a_1 to execute and that there are configurations of $\mathcal{A}$ where more than one rule, can execute besides a_1. By defining $\psi(W) = (R_a \setminus (W) \cup ((I_a \cup \bigcup_{i=2,\ldots,n}(R_{a_i} \setminus R_{a_1})) \cap W)$ only a_1 can be executed in any W.

5 Related work

To the best of our knowledge, this is the first paper which discusses different modalities of contribution from the environment. As such, a number of notions from the classical model could be adapted to the new model. In particular, the notion of causality [7] can be extended to account for the contribution from functional environments, so that the product influence and the resource dependence of a set of elements can now consider also the elements that the environment will provide, based on the resulting configuration. Due to Theorem 1, this modification is relevant only for the case where the environment is *non-BC*.

Functions have been considered in the framework of reaction systems under two respects. On the one hand, the functions computed by reaction systems [12] have been studied. That kind of analysis can be extended to include the composition with the environmental contribution. On the other hand, measurement functions have been defined to annotate configurations with the result of specific measures, leading to the introduction of a notion of time in reaction systems [13]. In this case, however, functions produce elements in a different domain, without directly contributing to the configuration.

A notion of structured environment is discussed in [8], where it is used to model duration, i.e. the persistence of an element for a certain number of steps, even if it is not sustained by any reaction. This allows the modeling of elements with a decay time. This mechanism is not directly realisable with our notion of functional environment, which only depends on the products of the last reaction step, but would require a stateful environment.

The role of the environment in the determination of system behaviours has been usually approached through assume/guarantee approaches, where one predicates properties of the behaviour under certain assumptions on the context. Techniques have been developed in different areas, and recently in the area of verification of concurrent systems [10].

The notion of channel is analogous to that of filters in networks of evolutionary processors [9, 11], where only strings which belong to specific languages can be communicated. From the discussion in Section 3.3, we have observed that functional environments can simulate the use of channels. Similarly, it has been shown in [5] that the position of filters, whether on edges or inside nodes, is irrelevant with respect to the computational power of such networks.

6 Conclusions and future work

We have presented a number of variants on the classical model of reaction systems, analysing different types of constraints on the environmental contribution.

We have shown that the notion of a functional environment not respecting boundary conditions introduces a new class of families of sequences, intermediate between the families of sequences from context-independent systems and those from systems with unconstrained environments. Moreover, the use of symmetrical difference to further constrain the contribution from a functional environment provides a powerful tool to control the evolution of a system.

The proposed mechanisms are motivated by the need to model situations in which a system is immersed in an environment, which is in turn influenced by the system state, and can exhibit reactive, collaborative, or adversarial behaviours.

We have not placed any specific restriction on the nature of the functions modeling the environment, so that future work will consider them. In particular, we have considered a stateless environment, in which the contribution only depends on the system configuration. It might be interesting to study the case where the contribution also depends on some environmental state, generalising the extension to durations in [8]. This leads naturally to a model where the

environment is in turn a network of reaction systems, communicating through channels. As hinted at in Section 5, the model could then also place restrictions on both sides of the communication channel.

Similarly, we plan to expand the study of the interplay between specific types of reaction set and specific types of environment contribution, that has been restricted here to the case of autodual systems.

7 Acknowledgements

We thank Grzegorz Rozenberg for invaluable input, pointing out the relevance of boundary conditions and suggesting to investigate other types of operations.

References

1. J.-M. Andreoli and R. Pareschi. Linear ojects: Logical processes with built-in inheritance. *New Generation Computing*, 9(3/4):445–474, 1991.
2. J.-P. Banâtre, P. Fradet, and D. L. Métayer. GAMMA and the chemical reaction model: Fifteen years after. In *Multiset Processing, Mathematical, Computer Science, and Molecular Computing Points of View*, volume 2235 of *LNCS*, pages 17–44. Springer, 2000.
3. J.-P. Banâtre and D. L. Métayer. The GAMMA model and its discipline of programming. *SCP*, 15(1):55–77, 1990.
4. G. Berry and G. Boudol. The Chemical Abstract Machine. *TCS*, 96(1):217–248, 1992.
5. P. Bottoni, A. Labella, F. Manea, V. Mitrana, and J. M. Sempere. Filter position in networks of evolutionary processors does not matter: A direct proof. In *Proc. DNA 15*, volume 5877 of *LNCS*, pages 1–11. Springer, 2009.
6. R. Brijder, A. Ehrenfeucht, M. G. Main, and G. Rozenberg. A tour of reaction systems. *IJFCS*, 22(7):1499–1517, 2011.
7. R. Brijder, A. Ehrenfeucht, and G. Rozenberg. A note on causalities in reaction systems. *ECEASST*, 30, 2010.
8. R. Brijder, A. Ehrenfeucht, and G. Rozenberg. Reaction systems with duration. In *Computation, Cooperation, and Life*, volume 6610 of *LNCS*, pages 191–202. Springer, 2011.
9. J. Castellanos, C. Martín-Vide, V. Mitrana, and J. M. Sempere. Networks of evolutionary processors. *Acta Inf.*, 39(6-7):517–529, 2003.
10. L. D'Errico and M. Loreti. Assume-guarantee verification of concurrent systems. In *Proc. Coordination 2009*, volume 5521 of *LNCS*, pages 288–305. Springer, 2009.
11. C. Drăgoi, F. Manea, and V. Mitrana. Accepting networks of evolutionary processors with filtered connections. *JUCS*, 13(11):1598–1614, 2007.
12. A. Ehrenfeucht, M. G. Main, and G. Rozenberg. Functions defined by reaction systems. *IJFCS*, 22(1):167–178, 2011.
13. A. Ehrenfeucht and G. Rozenberg. Introducing time in reaction systems. *TCS*, 410(4-5):310–322, 2009.
14. G. Paun. Membrane computing. In R. A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 5523–5535. Springer, 2009.
15. G. Paun, G. Rozenberg, and A. Salomaa. *DNA computing - new computing paradigms*. Texts in theoretical computer science. Springer, 1998.

The Nature of Computation and The Development of Computational Models

Mark Burgin[1] and Gordana Dodig-Crnkovic[2]

[1] Department of Mathematics, UCLA, Los Angeles, USA
mburgin@math.ucla.edu,
[2] School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
gordana.dodig-crnkovic@mdh.se

Abstract. The notion of computation is historically developing just like other fundamental nations, depending on both the development of related theory and on practical applications. Instead of providing a definite, once-and-for-all definition, we analyze the notion of computation and argue that we are far from complete understanding of what computation is and what it could be developed into. This paper presents a study in the nature of contemporary computation, contributing with computation typologies: essential, spatial, temporal, representational and hierarchy-level based. Drawing parallels with the historical development of the idea of number we argue that the concept of computation must necessarily develop. We thus address the development of models of computation, with emphasis on the development potential of natural/physical/embodied computation and unconventional computing. Our analysis suggests how better understanding of computation both as a model and as the underlying physical processes is needed than we have today. Finally, we indicate possible directions for future research.

1 Introduction

Many researchers have asked the question "What is computation?" trying to find a universal definition of computation or, at least, a plausible description of this important type of processes (cf. for example [1] [2] [3] [4] [5] [6]).

Some did this in an informal setting based on computational and research practice, as well as on philosophical and methodological considerations. Others strived to build exact mathematical models to comprehensively describe computation, and when the Turing machine was constructed and accepted as a universal computational model (referring to Church's endorsement), they imagined achieving the complete and exact definition of computation. However, the absolute nature of a Turing machine model as "The model of computation" to which all other possible models are equivalent, was disproved and in spite of all efforts, our understanding of computation remains too vague and ambiguous.

This vagueness of foundations has resulted in a variety of approaches, including approaches that contradict each other. For instance,Copeland [3] writes "to compute is to execute an algorithm." Active proponents of the Church-Turing Thesis, such as [7] claim computation is bounded by what Turing machines are doing. For them the problem of defining computation was solved long ago with the Turing machine model.

On the other hand, Wegner and Goldin insist that computation is an essentially broader concept than an algorithm [8] and propose an interactive view of computing. At the same time [9] argues that computation is symbol manipulation. Neuroscientists on the contrary describe sub-symbolic computation in neurons. [10]

Existence of various types and kinds of computation, as well as a variety of approaches to the concept of computation, shows the complexity of understanding of computation. To work out the situation, we analyzed historical developments in science and mathematics when attempts were made at finding comprehensive definitions of basic scientific and mathematical ideas.

For instance, mathematicians tried to define a number for millennia. However, all the time new kinds of numbers were introduced changing the comprehension of what a number is. Looking back we see that at the beginning, numbers came from counting and there was only a finite amount of numbers. Then mathematicians found a way to figure out the infinite set of natural numbers, constructing it with 1 as the building block and using addition as the construction operation. As 1 played a specific role in this process, for a while, mathematicians excluded 1 from the set of numbers. At the same time, mathematicians introduced fractions as a kind of numbers. Later they understood that fractions are not numbers but only representations of numbers. They called such numbers rational as they represented a rational, that is, mathematical, approach to quantitative depiction of parts of the whole. Then a number zero was discovered. Later mathematicians constructed negative numbers, integer numbers, real numbers, imaginary numbers and complex numbers. It looked like as if all kinds of numbers had been already found. However, the rigorous representation of complex numbers as vectors in a plane gave birth to diverse number-like mathematical systems and objects, such as quaternions, octanions, etc. Even now only few mathematicians regard these objects as numbers.

A little bit later, the great mathematician Cantor [11] introduced transfinite numbers, which included cardinal and ordinal numbers. So, the family of numbers was augmented by an essentially new type of numbers and this was not the end. In the 20th century, [12] introduced nonstandard numbers, which included hyperreal and hypercomplex numbers. Later [13] founded surreal numbers and [14] established hypernumbers, which included real and complex hypernumbers. This process shows that it would be inefficient to restrict the concept of a number by the current situation in mathematics. This history helps us also to come to the conclusion that it would be unproductive to restrict the concept of computation by the current situation in computer science and information theory.

In this paper, we present a historical analysis of the concept of computation before and after electronic computers were built and computer science emerged, demonstrating that history brings us to the conclusion that efforts in building such definitions by traditional approaches would be inefficient, while an effective methodology is to find essential features of computation with the goal to explicate its nature and to build adequate models for research and technology.

Consequently, we study computation in the historical perspective, demonstrating the development of this concept on the practical level related to operations performed by people and computing devices, as well as on the theoretical level where computation

is represented by abstract (mostly mathematical) models and processes. This allows us to discover basic structures inherent to computation and to develop a multifaceted typology of computations.

The paper is organized in the following way. In Section 2, we study the structural context of computation, explicating the Computational Triad and the Shadow Computational Triad. In Section 3, we develop computational typology, which allows us to extract basic characteristics of computation and separate fundamental computational types. The suggested system of classes allows us to reflect a natural structure in the set of computational processes. In Section 4 we present the development of computational models, and particularly natural computing. Finally, we summarize our findings in Section 5.

2 Structural Context of Computation

The first intrinsic structure, the Computational Dyad was introduced in [15], (Figure 1):

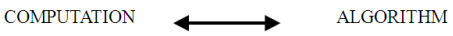


Fig. 1: The Computational Dyad

The Computational Dyad reflects the existing duality between computations and algorithms. According to [5], in the 1970s Dijkstra defined an algorithm as a static description of computation, which is a dynamic state sequence evoked from a machine by the algorithm. Later a more systemic explication of the duality between computations and algorithms was elaborated. Namely, computation is a process of information transformation, which is organized and controlled by an algorithm, while an algorithm is a system of rules for a computation [4]. In this context, an algorithm is a compressed informational/structural representation of a process.

Note that a computer program is an algorithm written in (represented by) a programming language. This shows that an algorithm is an abstract structure and it is possible to realize one algorithm as different programs (in different programming languages). Moreover, many people think that neural networks perform computations without algorithms. However, this is not true because neural networks algorithms have representations that are different from traditional representations of algorithms as systems of rules/instructions. The neural networks algorithms are represented by neuron weights and connections between neurons. This is similar to hardware representation/realization of algorithms in computers (analog computing). However, the Computational Dyad is incomplete because there is always a system that uses algorithms to organizes and controls computation. This observation shows that the Computational Dyad has to be extended to the Computational Triad (cf. Figure 2).

Note that the computing device can be either a physical device, such as a computer, or an abstract device, such as a Turing machine, or a programmed (virtual or simulated) device when a program simulates some physical or abstract device. For instance, neural networks and Turing machines are usually simulated by programs in conventional

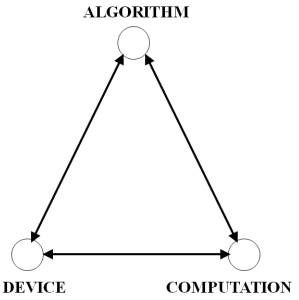


Fig. 2: The Computational Triad

computers. Or Java virtual machine can be run on different operating systems and is processor and operating system independent.

Besides, with respect to architecture, it can be an embracing device, in which computation is embodied and exists as a process, or an external device, which organize and control computation as an external process. It is also important to understand the difference between algorithm and its representation or embodiment. An algorithm is an abstract structure, which can be represented in a multiplicity of ways: as a computer program, a control schema, a graph, a system of cell states in the memory of a computer, a mathematical system, such as an abstract finite automaton, etc.

In addition, there are other objects essentially related to computation. Computation always goes in some environment and within some context. Computation always works with data, performing data transformations. Besides, it is possible to assume that computation performs some function and has some goal (for some agent) even if we don't know this goal. The basic function of computation is information processing. These considerations bring us to the Shadow Computational Triad (cf. Figure 3).

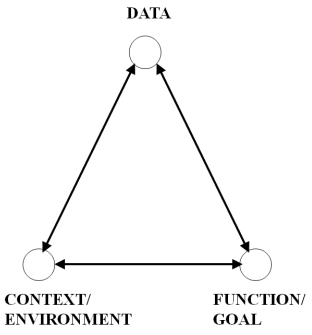


Fig. 3: The Shadow Computational Triad

Thus, the Shadow Computational Triad complements the Computational Triad reflecting that any computation has a goal, goes on in some context, which includes environment, and works with data. In a computation, information is processed by data transformations.

3 Computational Typology

There are many types and kinds of computations utilized by people and known to people. The structure of the world [16] implies the following classification.

3.1 Existential/substantial typology of computations

According to the substrate carrying on computation, we distinguish the following types:

- Physical or embodied computations.
- Abstract or structural computations.
- Mental or impersonalized computations.

In agreement with contemporary science, abstract and mental computations are always based on some embodied computations. The difference is on the level of organization/level of abstraction. The existential types from this typology have definite subtypes, as follows.

- Physical or embodied computations.
  1. Biological computations.
  2. Chemical computations.
  3. Technological computations.

- Structural/abstract computations.
  1. Symbolic computations.
  2. Subsymbolic computations.
  3. Iconic computations.

There are connections between these types. For instance, as [17] suggests, the principle of object formation may be an example of the transition from a stream of massively parallel subsymbolic microfunctional events to symbol-type, serial processing through subsymbolic integration. In addition to the existential typology, there are other typologies of computations.

3.2 Spatial typology of computations

According to the spatial aspect, computations can be categorized in the following way.

1. Centralized computations where computation goes controlled by a single algorithm.
2. Distributed computations where there are separate algorithms that control computation in some neighbourhood. Usually a neighbourhood is represented by a node in the computational network.

3. Clustered computations where there are separate algorithms that control computation in clusters of neighbourhoods.

Turing machines, partial recursive functions and limit Turing machines are models of centralized computations. Neural networks, Petri nets and cellular automata are models of distributed computations. Grid automata in which some nodes represent networks with the centralized control [4] and the World Wide Web are systems that perform clustered computations.

3.3 Temporal typology of computations

1. Sequential computations, which are performed in linear time.
2. Parallel or branching computations, in which separate steps (operations) are synchronized in time.
3. Concurrent computations, which do not demand synchronization in time.

Note that while parallel computation is completely synchronized, branching computation is not completely synchronized because separate branches acquire their own time and become synchronized only in interactions.

3.4 Representational or operational typology of computations

1. Discrete computations, which include interval computations.
2. Continuous computations, which include fuzzy continuous computations.
3. Mixed computations, some include discrete and continuous processes.

Digital computer devices and the majority of computational models, such as finite automata, Turing machines, recursive functions, inductive Turing machines, and cellular automata, perform discrete computations. Examples of continuous computations are given by abstract models, such as general dynamical systems [18] and hybrid systems [19], and special computing devices, such as the differential analyzer [20] [21]. Mixed computations include piecewise continuous computations, combining both discrete computation and continuous computation. Examples of mixed computations are given by neural networks [22], finite dimensional machines and general machines of [23].

3.5 Hierarchy of levels of computation

In [6] three generality levels of computations are introduced.

1. On the top and most abstract/general level, computation is perceived as any transformation of information and/or information representation.
2. On the middle level, computation is distinguished as a discretized process of transformation of information and/or information representation.
3. On the bottom, least general level, computation is recognized as a discretized process of symbolic transformation of information and/or symbolic information representation.

There are also spatial levels or scales of computations, according to the size of computational devices:

1. The macrolevel includes computations performed by macroscopic systems such as electromechanical devices, vacuum tubes and/or transistors, as well as mechanical calculators.
2. The microlevel includes computations performed by integrated circuits.
3. The nanolevel includes computations performed by fundamental parts that are not bigger than a few nanometers.
4. The molecular level includes computations performed by molecules.
5. The quantum level includes computations performed by atoms and sub-atomic particles.

There are no commercially available nanocomputers, molecular or quantum computers in existence at present. However, current chips produced by nanolithography are close to computing nanodevices because their basic elements are less than 100 nanometers in scale.

4 Natural Computing

The development of computing, both machinery and its models, continues. We are used to quick increase of computational power, memory and usability of our computers, but the limit of miniaturization is approaching as we are getting close to quantum dimensions of hardware. One of the ideals of computing ever since time of Turing is intelligent computing which would besides mechanical include even intelligent problem solving. Currently, there is a development of cognitive computing aimed towards human abilities to process/organize/understand information.

At the same time development of computational modelling of human brain has for a goal to reveal the exact mechanisms of human brain function that will help us understand not only how humans actually perform symbol processing when they follow an algorithm, but also how humans in the first place create algorithms or models. Those new developments can be seen as a part of the research within the field of natural computing, where natural system performing computation is human brain.

However, natural computing has a much broader scope. According to the Handbook of Natural Computing [24] natural computing is ”the field of research that investigates both human-designed computing inspired by nature and computing taking place in nature.” It addresses both computational models inspired by the natural systems, computation performed by natural materials and computational nature of processes taking place in (living) nature. It includes among others areas of cellular automata and neural computation, evolutionary computation, molecular computation and quantum computation and nature-inspired algorithms and alternative models of computation.

An important characteristic of the research in natural computing is that knowledge is generated bi-directionally, through the interaction between computer science and natural sciences. While the natural sciences are rapidly absorbing ideas, tools and methodologies of information processing, computer science is broadening the notion of computation, recognizing information processing found in nature as natural computation.

[25] [26] [27] That development led Denning [28] to argue that computing today is a natural science. We can add that natural sciences today are permeated by computing in forms of computational tools, models and conceptualizations.

This new concept of computation with inspiration in natural information processing allows among others learning about non-deterministic complex computational systems with self-* properties (self-organization, self-configuration, self-optimization, self-healing, self-protection, self-explanation, and self-awareness. Through its layered architecture of computational processes, natural computation provides a basis for a unified understanding of phenomena of embodied cognition, intelligence and knowledge generation. [29] [30]

In addition to the current developments within natural computing with its emphasis on physical computation, there is an important development in the field of unconventional algorithms. It is worth to notice the complementarity of axiomatics and construction, as emphasized in [31] for the case of unconventional algorithms, both elements being necessary for the progress in our understanding of computation. Here new powerful tools are brought forth by local mathematics, local logics, logical varieties and the axiomatic theory of algorithms, automata and computation. Further work includes study of natural computation by unconventional algorithms and constructive approaches.

5 Conclusions and Future Work

In order to bring more structure in the current discussion about the nature of computation we give an account of computation typologies: essential, spatial, temporal, representational and hierarchy-level. The historical development of the idea of number suggests that the concept of computation as well historically develops and we indicate current directions of development. We find that especially promising steps forward are natural computing and unconventional computing.

Among the issues for the future research, there are numerous open problems related to the nature of information and computation, as well as to their relationships. How is information dynamics represented in computational systems, in machines, in living organisms and in the physical world in general? How to relate natural computational processes with information systems, knowledge management and understanding of intelligence?

References

1. Turing, A.M.: On computable numbers, with an application to the entscheidungs problem. Proceedings of the London Mathematical Society **42**(42) (1936) 230–265
2. Kolmogorov, A.: On the concept of algorithm. Russian Mathematical Surveys **8**(4) (1953) 175–176
3. Copeland, B.: What is computation? Synthese **108**(3) (1996) 335–359
4. Burgin, M.: Super-Recursive Algorithms. Springer New York Inc. (2005)
5. Denning, P.: What is computation? Editors introduction. Ubiquity (October) (2010) 1–2
6. Burgin, M., Dodig-Crnkovic, G. In: Information and Computation - Omnipresent and Pervasive. World Scientific Pub Co Inc (2011) vii–xxxii

7. Fortnow, L.: The enduring legacy of the Turing machine. The Computer Journal **55**(7) (2012) 830–831
8. Goldin, D., Smolka, S., Wegner, P.: Interactive Computation: The New Paradigm. Springer (2006)
9. Conery, J.: Computation is symbol manipulation. The Computer Journal **55**(7) (2012) 814–816
10. Angelaki, D., Shaikh, A., Green, A., Dickman, J.: Neurons compute internal models of the physical laws of motion. Nature **430**(6999) (2004) 560–564
11. Cantor, G.: ber unendliche lineare Punktmannigfaltigkeiten, 5. Grundlagen einer allgemeinen Mannigfaltigkeitslehre. Ein mathematisch-philosophischer Versuch in der Lehre des Unendlichen. Teubner (1883)
12. Robinson, A.: Non-standard analysis. Indagationes Mathematicae **23** (1961) 432–440
13. Conway, J.: On numbers and games. Academic Press (1976)
14. Burgin, M.: Hypermeasures and hyperintegration. Notices of the National Academy of Sciences of Ukraine **6** (1990) 10–13
15. Burgin, M., Eberbach, E.: Evolutionary computation and processes of life. Ubiquity (August) (2012) 1–13
16. Burgin, M.: Structural Reality. Nova Science Publishers (2012)
17. Bucci, W.: Psychoanalysis and Cognitive Science: A Multiple Code Theory. (1997)
18. Bournez, O.: Achilles and the tortoise climbing up the hyper-arithmetical hierarchy. Theoretical Computer Science **210** (1977) 210–211
19. Gupta, V., Jagadeesan, R., Saraswat, V.: Computing with continuous change. Science of Computer Programming **30**(1-2) (1998) 3–49
20. Shannon, C.: Mathematical theory of the differential analyzer. J. Math. Physics **20** (1941) 337–354
21. Moore, C.: Recursion theory on the reals and continuous-time computation. Theoretical Computer Science **162**(1) (1996) 23–44
22. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. 1943. Bulletin of Mathematical Biology **52**(1-2) (1990) 99–115
23. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and real computation: A manifesto. International Journal Of Bifurcation And Chaos **6**(1) (1996) 3–26
24. Rozenberg, G., Bäck, T., Kok, J.N.: Handbook of Natural Computing. Springer (2012)
25. Rozenberg, G., Kari, L.: The many facets of natural computing. Communications of the ACM **51** (2008) 72–83
26. Stepney, S., Braunstein, S.L., Clark, J.A., Tyrrell, A.M., Adamatzky, A., Smith, R.E., Addis, T.R., Johnson, C.G., Timmis, J., Welch, P.H., et al.: Journeys in non-classical computation i: A grand challenge for computing research. Int. J. Parallel Emerg. Distr. Syst. **20** (2005) 5–19
27. Stepney, S., Braunstein, S.L., Clark, J.A., Tyrrell, A.M., Adamatzky, A., Smith, R.E., Addis, T.R., Johnson, C.G., Timmis, J., Welch, P.H., et al.: Journeys in non-classical computation ii: Initial journeys and waypoints. Int. J. Parallel Emerg. Distr. Syst. **21** (2006) 97–125
28. Denning, P.: Computing is a natural science. Communications of the ACM **50**(7) (2007) 13–18
29. Dodig-Crnkovic, G., Mueller, V. In: A Dialogue Concerning Two World Systems: Info-Computational vs. Mechanistic. Volume 2 of World Scientific Series in Information Studies. Springer (2010) 149–84
30. Wang, Y.: On abstract intelligence: Toward a unifying theory of natural, artificial, machinable, and computational intelligence. Int. J. of Software Science and Computational Intelligence **1**(1) (2009) 1–17
31. Dodig-Crnkovic, G., Burgin, M.: Unconventional algorithms: Complementarity of axiomatics and construction. Entropy **14**(11) (2012) 2066–2080

TOWARDS A CHURCH-TURING-THESIS FOR INFINITARY COMPUTATIONS

MERLIN CARL

Abstract. We consider the question whether there is an infinitary analogue of the Church-Turing-thesis. To this end, we argue that there is an intuitive notion of transfinite computability and build a canonical model, called Idealized Agent Machines (IAMs) of this which will turn out to be equivalent in strength to the Ordinal Turing Machines defined by P. Koepke.

1. Introduction

Since [ITTM], various generalizations of classical notions of computability to the transfinite have been given and studied. The Infinite Time Turing Machines ($ITTM$s) of Hamkins and Lewis generalized classical Turing machines to transfinite working time. Ordinal Turing Machines (OTMs) (see [OTM]) and Ordinal Register Machines (ORMs) further generalized this by allowing working space of ordinal size. Recently, a transfinite version of λ-calculus was introduced and studied [Sey]. It was soon noted (see e.g. [Fi]) that the corresponding notion of computability enjoys a certain stability under changes of the machine model: For example, the sets of ordinals computable by OTMs and ORMs both coincide with the constructible sets of ordinals.

A similar phenomenon is known from the models of classical computability: Turing machines, register machines, recursive functions, λ calculus etc. all lead to the same class of computable functions. In the classical case, this is taken as evidence for what is known as the Church-Turing-Thesis (CTT), i.e. the claim that these functions are exactly those computable in the 'intuitive sense' by a human being following a rule without providing original input. This thesis plays an important role in mathematics: It underlies, for example, the - to our knowledge undisputed[1] - view that Matiyasevich's theorem [Ma] settles Hilbert's 10th problem or that Turing's work [Tu1] settles the Entscheidungsproblem. The study of recursive functions gets a lot of its attraction from this well-grounded belief that they coincide with this intuitive notion of computability.

It therefore seems natural to ask whether something similar can be said about transfinite models of computation, i.e. whether these models are mere 'ordinalizations' of the classical models or whether they actually 'model' something, whether there is an intutive concept of transfinite computability that is captured by these models: Hence, we ask for an infinitary Church-Turing-thesis ($ICTT$).
There seems to be some evidence that a satisfying $ICTT$ should be obtainable. Beside the stability of the corresponding notion of computability mentioned above, it also became common to describe and communicate the activity of such machines in rather informal terms: Rather than writing an actual program for e.g. deciding

[1] It has been remarked that there are challenges to the claim that no physical device could decide such questions, see e.g. [Hog] and [NeGe]. However, here we are interested in the capabilites of idealized computing agents. Whether what such devices do can be considered to be a computation in the intuitive sense rather than the observation of an incomputable process is a question we won't consider here.

number-theoretical statements with an $ITTM$, it generally suffices to explain that the machine will e.g. 'search through the naturals for a witness'. It usually soon becomes clear to someone with a basic familiarity with these models that such a method can indeed be implemented and will lead to the right results. Indeed, we will usually find such a 'process description' much easier to grasp than an actual implementation. This indicates that we indeed possess an intuitive understanding of what these machines can do which is based on an understanding of infinite processes rather than the formal definition of the machine. We aim at connecting infinitary models of computation with a natural notion. Here, 'natural' means that the notion can be obtained and described independently from the models and that it is in some sense present in normal (mathematical) thinking. Such a notion should furthermore serve as a background thesis explaining the equivalence of the different models, should (in analogy with the classical Church-Turing-thesis) justify the use of informal 'process descriptions' to prove the existence of formally specified programs and, ideally, allow mathematically fruitful applications, similar to the role the classical CTT plays in e.g. Hilbert's 10th problem.
In this work, we offer evidence for the claim that notions of transfinite computation are indeed naturally present in mathematical (and possibly in everyday) thinking and that these notions are captured by the transfinite machine models we mentioned.[2] This will allow us to formulate an $ICTT$.

This article is structured as follows: We begin by describing an approach of mathematical philosophy initiated by P. Kitcher [Ki], where mathematical objects are modelled as mental constructions of idealized agents. We also indicate that such idealizations are indeed present in understanding mathematics. After that, we work towards a formal notion of a computing transfinite agent, obtaining the notion of an Idealized Agent Machine (IAM). Then, we show that the computational power of an IAM coincides with that of OTMs and ORMs (which we will summarize under the term 'standard models' from now on). Finally, we state (a candidate for) an $ICTT$ and discuss whether it meets the above requirements.

2. Idealized Constructions and Idealized Agents in Mathematics

In this section, we briefly describe the view on the philosophy of mathematics described in [Kitcher]. We use his account as a demonstration that the concept of transfinite agents can be motivated and has arisen completely independent from our considerations. Furthermore, we want to indicate how these views can be fruitful for infinitary computations (and vice versa) and bring them into interaction. Finally, his work serves us as a first introduction to the notion of idealized agents. We will then demonstrate that this notion seems indeed to be present in mathematical language and understanding.

2.1. **Kitcher's idealized-agents-view of mathematics.**

In a nutshell, Kitcher attempts to justify an empiricist account of mathematics by describing mathematics as an idealization of operations with real-world objects like grouping them together, adding an object to a pile of objects etc. These actions in themselves already are a kind of primitive mathematics, limited by our practical constraints. What is usually called mathematics is obtained by forming a theory of idealized operations in a similar way that, say, a theory of idealized gases is formed: We abstract away from certain 'complicating factors' like e.g. our factual incabability of indefinitely adding objects to a collection. Mathematics is then the study of idealized operations, or, equivalently, of the operations of idealized agents.

[2] To be precise, we will argue for this claim in the case of OTMs and ORMs. Whether similar approaches are available for other models as well is briefly adressed at the end of this paper.

Upon reading this, one might wonder how this account is supposed to make sense of the large parts of mathematics which, like axiomatic set theory, deal with actual infinite objects. Kitcher's reply to this is simply that this is a mere question of the degree of idealization:

> [Ki], p. 146: I see no bar to the supposition that the sequence of stages at which sets are formed is highly superdenumerable, that each of the stages corresponds to an instant in the life of the constructive subject, and that the subject's activity is carried out in a medium *analogous* to time, but far richter than time. (Call it 'supertime'.) ... The view of the ideal subject as an idealization of ourselves does not lapse when we release the subject from the constraints of our time.

Comparing Kitcher's account of axiomatic set theory with his treatment of arithmetic or intuitionistic mathematics, mathematical areas can roughly be characterized by the degree of idealization, i.e. by considering how remote the underlying operations are from our actual capabilities. The agent working in 'supertime' mentioned in the quote above seems to belong to a benchmark of idealization. As this is the degree of idealization corresponding to set theory in Kitcher's account, we will refer to it as the 'idealized agent of set theory' from now on.[3]

Not unexpectedly, several issues with this approach can and have been raised: E.g. about the ontological status of these idealized agents (discussed in [Ho]), whether this degree of idealization still admits an explanation of the applicability of mathematics, whether and how certain large cardinals can be accomodated in this account etc. Nevertheless, the imagination of an idealized agent or an idealized mental activity seems to be in the background of large parts of mathematical understanding in one way or the other. In fact, there are numerous common figures of speech in mathematical textbooks and even more in spoken conversation that point to such (implicit) notions: For example, in many proofs of the Bolzano-Weierstraß-theorem, 'we' are supposed to 'pick' a number from a subintervall containing infintely many elements of a given sequence. One might find this problematic: In a naive sense, of course, we cannot do this, as in general, we will not know which intervall that is.[4] However, this problem doesn't seem to come up in understanding this proof. In fact, agent-based formulations generally seem to increase understanding and make constructions more imaginable rather than leading into conflicts with our factual limitations. A similar observation holds for e.g. proofs of the well-ordering principle from the axiom of choice, and in general for many uses of transfinite recursion or transfinite induction. Another example would be the various places in mathematical logic where constructions are explained by interpreting them as transfinite 'games' between two 'players'.

2.2. **Degrees of idealization and the Church-Turing-Thesis.** In the Church-Turing-Thesis, recursiveness is stated to capture the intuitive meaning of 'computable'. However, if the intuitive meaning of 'computable' is taken as 'possible for a human being working without understanding', then literally, this is of course false: What we can actually do is very limited: In general, a recursive function is far away from being computable by 'a man provided with paper, pencil, and rubber, and subject to strict discipline' ([Tu]). But this fact is quite irrelevant for e.g.

[3]Similar ideas are mentioned in other accounts on the philosophy of mathematics. For example, in [Wa], S. 182, we find the following: 'The overviewing of an infinite range of objects presupposes an infinite intuition which is an idealization. Strictly speaking, we can only run through finite ranges (and perhaps ones of rather limited size only).'

[4]This is the reason why Bolzano-Weierstrass is intuitionistically invalid.

Hilbert's 10th problem, which asks for a 'finite' procedure, not a practical one. In the CTT, we are hence in fact facing a notion of an idealized computing subject.

Usually, this idealization goes from certain factual bounds to 'arbitarily large, but finite'. But there seems to be a distinguished intuitive notion of computability going beyond this: For example, there is little to no trouble with the idea of testing all even numbers for being a sum of at most two primes. In fact, this thought experiment seems to be at least part of the reason the Goldbach conjecture is generally assumed to have a definite truth value. On the other hand, no such intuition supports the idea of e.g. searching through V looking for a bijection between $\mathbb{R}$ and $\aleph_1$, not even if one assumes CH to have a definite truth value.[5] The idea of a transfinite systematic procedure for obtaining certain objects or answering certain questions hence allows for a clear distinction: Not every formulation that at the surface looks like a 'process description' is eligible as an indication of a computation of an idealized agent. Our goal is to find an exact characterization of those procedures that are.

3. A model for idealized Agents

Even if one accepts that, beyond finiteness, clear degrees of idealization of our activity can be concretely captured, the standard models are not as canonical a model of it as e.g. Turing machines are in the finite case. In the one direction, it does indeed seem plausible that the actions of an OTM are available to a transfinite idealized agent and that hence everything computable by an OTM should be computable by such an agent: The aspects of an OTM-computation going beyond classical computability consist in elementary limit operations like forming the limes inferior of a sequence of 0s and 1s. But the other direction is not as clear: For example, the limit rule of OTMs seems to be rather arbitrary. The intuition here is that other reasonable choices of limit rules will not change the class of computable objects, but it is exactly the intuition leading there that we want to capture here. We see no direct path from idealized agents to the standard models known so far. Our approach is hence to develop a formal notion of a transfinitely computing agent modelled after our intuition and then see how it relates to the standard models. It turns out that it does indeed describe the same notion of computability, which we consider a good piece of evidence for our thesis.

The notion we are about to develop will be called Idealized Agent Machines (IAMs). IAMs are meant to give a very liberal account of the computational activity of idealized agents. In fact, one might get the impression that what we model as a single step of an IAM is really a series of lengthy sub-computations and that we are hence far to generous in attributing abilities to our idealized agent. However, we will demonstrate that even this liberal notion is equivalent to the standard models. Therefore, we don't need to claim that IAMs are a very accurate model for the intuition of transfinite computations: we only need it to be strong enough to include that intuition. We can then argue that if such an intuition is really present - as we tried to show above, then it is grasped by the standard models, as, in the end, we will arrive at the following implications:

$$\begin{array}{c} OTM\text{-computable} \\ \underset{(1)}{\Longrightarrow} \text{ computable by the idealized agent of set theory} \\ \underset{(2)}{\Longrightarrow} IAM\text{-computable} \\ \underset{(3)}{\Longrightarrow} OTM\text{-computable} \end{array}$$

[5]Searching through L or its stages, on the other hand, seems again quite reasonable, as L is canonically well-ordered.

Here, implication (3), being a claim about two notions expressable in the language of set theory, is provable (in ZFC) and implication (1) is very natural (see above). It is step (2) that depends on the plausibility of the analysis and modelling we are about to give.

An ideal computing agent works as follows: At each time, he has a complete memory of his earlier computational activity. Also, he has a working memory where he may store information. We assume that the working memory consists of separate 'places', each containing one symbol from a finite alphabet.[6]

The agent is working in according with instructions that determine his activity. Certainly, any kind of operation that can be considered an idealization of an activity we are actually capable of must be describable by finite means. We hence stipulate that the instructions are given by some finite expressions. Based on the instructions, it must be possible at each time to determine what to do (e.g. which new symbols to write) on the basis of the computational activity so far. We propose to model this in the following way: There should be a first-order formula $\phi(x, y)$ such that, if the computational activity so far is given by c and p is a place in the memory, $\phi(c, p, s)$ holds iff s is the the symbol that should be written in place p after c. Here, it must be possible to evaluate ϕ by mere inspection of c. Even if 'inspection' may be taken in an idealized sense here as well, this should certainly mean that the appearing quantifiers should in some sense be 'bounded' by c. We will make this precise below.[7]
This description does not depend on any assumptions on the structure of time. It is hence sufficiently general to yield a notion of transfinite computability once an appropriate notion of transfinite time is introduced.

3.1. **Supertime and Superspace.** In the passage quoted in the first paragraph, Kitcher suggests that set theory can be considered as the outcome of the mental activity of an idealized agent working in 'a medium analogous to time, but far richer than time'. Here, we want to argue that the only sensible choice for such a medium are ordinals. In his argumentation, it is also implicitely assumed that the agent not only has a non-standard working time, but also the ability to 'store' the outcome of his work, e.g., infinite memory or at least infinite writing space. We will argue that it is natural and harmless to assume that the writing space of an idealized agent is indexed by ordinals.

Certainly, we intend a notion of time as a medium of a deterministic computation to be a linear ordering. But we can say more. The computational activity has to start at some point. Every other state may depend on this earlier state and hence has to take place at a moment after the starting point. Hence, the 'medium of computation' has to have a unique minimal element.

Whenever the agent has carried out a certain amount of computational activity, he has to know what to do next, i.e. there must be a unique next state for him to assume. This next state has to take place at some point of time. Hence, the medium in which he computes has to contain a unique next element after those through which the activity passed so far. Put differently: For every initial segment of time, there has to be a unique time point preceeded by all moments in the initial segment and only by those.

[6]The finiteness of the alphabet could in fact be dropped without changing the class of computable functions we ultimately obtain. However, we consider this a reasonable assumption for the notion we are about to model and hence decided against taking the effort to demonstrate this.

[7]The choice of first-order logic might be objectional; we feel that e.g. second-order logic would be inappropriate, for it would require the agent to have access to an external notion of set which is not determined from his computational activity. However, we are certainly interested in plausible alternatives and whether they would turn out to lead to an equivalent notion of computability.

This leads to the following notion of 'supertime': A 'supertime' is a linearly ordered set[8] $(X, \leq)$ with a unique minimal element μ and such that, for every proper initial segment I of X, there is a $\leq$-minimal $x_I \in X$ such that $\forall t \in I t < x_I$. It is now easy to see that this means that all candidates for supertime are (isomorphic to) ordinals:

Proposition 1. Let $(X, \leq)$ be a linearly ordered set such that, for every $I \subsetneq X$ which is downwards closed (i.e. $x < y \in I$ implies $x \in I$), there is a minimal $x_I \in X$ such that $\forall t \in I t < x_I$. Then $(X, \leq)$ is isomorphic to an ordinal.

Proof. Note that $\emptyset$ is downwards closed in $(X, \leq)$ and let $\mu := x_\emptyset$. Obviously, μ is the unique minimal element of X.
Let $A \subseteq X$. Consider the set $Y := \{x \in X | x < A\}$. It is easy to see that Y is an initial segment of X. We claim that x_Y is a minimal element of A.
To see that $x_Y \in A$, assume otherwise. As every element smaller than x_Y is in Y and hence smaller than every element of A, it follows that $x_Y < A$. But this implies $x_Y \in Y$, so $x_Y < x_Y$, a contradiction. So $x_Y \in A$ and every $z < x_Y$ satisfies $z \notin A$. Thus x_Y is indeed a minimal element of A. As $\leq$ is linear, x_Y is unique with this property.
This implies that $(X, \leq)$ is a well-ordered set. Hence, it is isomorphic to an ordinal. □

However, not all ordinals are suitable as such a medium: For example, if our medium allows two procedures to be carried out, it should also allow to carry out one after the other. Also, it should be possible to have a procedure as a 'subroutine' of another to be repeatedly called by the other. Finally, the class of ordinals itself provides an attractive unification of appropriate computation times; hence we allow computations carried out without fixing a particular ordinal in advance.
Appropriate candidates for supertime hence turn out to be ordinals which are closed under ordinal addition and multiplication and On itself. In the following, we will - for the sake of simplicity - focus on the broadest case where the underlying time is On. Note that this notion of supertime matches well with the way transfinite constructions are commonly communicated and imagined: It is completely normal to relate stages of such a construction by expressions coming from the relation of time points and state that e.g. 'earlier on, we made sure that'. In fact, it is hard to talk about transfinite constructions avoiding such expressions.

We imagine our agent to be equipped with a sufficient supply of place for writing symbols. We assume that this space is partioned into slots and that each slot is uniquely recognizable. There is a canonical well-ordering on the set of used slots: Namely, each slot is at some point of time used for the first time. Via this property, this slot is henceforth identifiable. We may hence assume for our convenience that the slots are indexed with ordinals from the very beginning: That is, the working memory is at any time a function from some ordinal α into the set S of symbols.[9]

Finally, even if we allow - as we will - several symbols to be re-written in one step, an adequate model of computing time and space should also impose some bounds on the space that can be actually used after computing for τ many steps. We model this intuition by the extra condition that, at time τ, only slots with index in τ may contain written symbols.[10]

[8]The outcome might be different if one would allow 'class time'. We don't pursue this further here.

[9]This point could be strengthened by modelling space in a more general way and then proving the resulting notion to be equivalent with the one obtained here. However, this requires a cumbersome analysis and the gain in plausibility seems to be too limited to justify it.

[10]This condition may seem to be too strict compared to the overall very liberal model we set up. However, this choice is technically the least cumbersome; furthermore, we conjecture from

3.2. **Idealized Agent Machines.** We will now describe a formal model for the concept developed above. The instructions will be given by a first-order statement in an appropriate language, which can be evaluated on the basis of an initial segment of a computation.

We let L_c be the first-order language with equality, a binary function symbol $C(x,y)$ and a binary relation symbol $\leq$. The intended meaning of $C(x,y)=z$ is that, at time x, z is the symbol in the yth place, while $\leq$ is the ordering relation of ordinals.

If A is a finite set (the alphabet) and τ an ordinal, then a τ-state for A is a function $f:\alpha\to A$, where $\alpha\leq\tau$. We denote the class of τ-states for A by S^τ_A.

A function F with $dom(F)=:\tau\in On$ and $F(\iota)\in S^\iota_A$ for all $\iota<\tau$ is called an A-τ-precomputation. For F an A-τ-precomputation, an L_c-formula ϕ, $\vec{s}\in A^{<\omega}$, $\vec{\alpha}\in(\tau+1)^{<\omega}$, we define $[\phi(\vec{\alpha},\vec{s})]^F_\tau$, the truth value of $\phi(\vec{\alpha},\vec{s})$ in F, recursively (omitting the parameters where possible): $[C(\alpha,\beta)=x]^F_\tau=1$ if $\alpha<\beta$ or $F(\alpha)(\beta)=x$, otherwise $[C(\alpha,\beta)=x]^F_\tau=0$; $[x\leq y]^F_\tau=1$ iff $x,y\in On$ and $x\leq y$, otherwise $[x\leq y]^F_\tau=0$; $[x=y]^F_\tau=1$ iff $x=y$, otherwise $[x=y]^F_\tau=0$; $[\neg\phi]^F_\tau=1-[\phi]^F_\tau$; $[\phi\wedge\psi]^F_\tau=[\phi]^F_\tau[\psi]^F_\tau$; and $[\exists x\phi(x)]^F_\tau=1$ iff there is $\iota\in\tau$ such that $[\phi(\iota)]^F_\tau=1$, otherwise $[\exists x\phi(x)]^F_\tau=0$.

An L_c-formula $\phi(x,y,z)$ is an IAM-program iff, for all $\tau\in On$, $\alpha\leq\tau$ and all A-τ-precomputations F, there is exactly one $s\in A$ such that $[\phi(\tau,\alpha,s)]^F_\tau=1$. If ϕ is an IAM-program, A a finite set, $\tau\in On$ and F an A-τ-precomputation, then we define $\mathbb{S}_{\phi,\tau,F}:\tau\to A$, the state of the IAM-computation with ϕ at time τ after F, by letting $\mathbb{S}_{\phi,\tau,F}(\alpha)$ be the unique $s\in A$ such that $\phi(F,\alpha,s)$ holds for $\alpha<\tau$.

Furthermore, we define $\mathbb{I}^\tau_\phi$, the τ-th initial segment of the IAM-computation with ϕ at time τ, recursively by letting $\mathbb{I}^0_\phi:=\emptyset$, $\mathbb{I}^{\tau+1}_\phi:=\{(\tau,\mathbb{S}_{\phi,\tau,\mathbb{I}^\tau_\phi})\}\cup\mathbb{I}^\tau_\phi$ and $\mathbb{I}^\lambda_\phi:=\bigcup_{\iota<\lambda}\mathbb{I}^\iota_\phi$ for λ a limit ordinal.

So far, our machines have no notion of halting. We therefore assume that all our IAMs have a special symbol $\mathbb{H}$ in their alphabet. The IAM-computation by ϕ is said to have stopped at time τ iff $\mathbb{I}_{\phi,\tau}(\tau)(0)=\mathbb{H}$, i.e. if the first symbol in the memory at time τ is $\mathbb{H}$.

An IAM-computation by ϕ will hence start with an empty tape and then repeatedly apply the $\mathbb{S}$-operator to obtain the next state, taking unions at limits.

It is easy to see from the boundedness of the formula evaluated in each step that this notion of computability is absolute insofar IAM-computations are absolute between transitive models of ZFC. We can also account for computations with a non-empty input and computations with parameters in these terms by adjusting the initial memory content.

Definition 2. $X\subseteq On$ is IAM-computable iff there exists an IAM-program ϕ such that, for every $\alpha\in On$, there is $\tau\in On$ such that, if χ_α is the characteristic function of α in On and $F=(0,\chi_\alpha)$, we have $\mathbb{S}_{\phi,\tau,F}(0)=\mathbb{H}$ and $\mathbb{S}_{\phi,\tau,F}(1)=1$ iff $\alpha\in On$.

Similarly, $f:On\to On$ is IAM-computable iff there is an IAM-program ϕ such that, for every $\alpha\in On$, there is $\tau\in On$ such that $\mathbb{S}_{\phi,\tau,F}(0)=\mathbb{H}$, $\mathbb{S}_{\phi,\tau,F}(f(\alpha)+1)=1$ and $\mathbb{S}_{\phi,\tau,F}(\iota)=0$ for $\iota\notin\{0,f(\alpha)+1\}$, where again $F=(0,\chi_\alpha)$ and χ_α is the characteristic function of α in On.

We say that a set $X\subseteq On$ or a function $f:On\to On$ is IAM-computable from finitely many ordinal parameters iff there exists a finite set $p\subset On$, an IAM-program ϕ using the alphabet A and an $a\in A$ such that ϕ computes X (or f,

---

our experience so far that every bound that is reasonably explicit in τ will ultimately lead to the same class of computable functions.

respectively) when the following change is made for all $\tau < \alpha \in On$ in the definition of the α-th state $\mathbb{S}_{\phi,\alpha,\mathbb{I}^\alpha_\phi}$: If $\beta \in p$, then $\mathbb{S}_{\phi,\alpha,\mathbb{I}^\alpha_\phi}(\beta)$ is set to a.

4. Idealized Agent Machines, ordinal computability and the $ICTT$

Having developed our formal model for infinitary computations, it is now rather straightforward to show that, in terms of computability, it is equivalent to the standard models. As the elobarate versions are quite long and cumbersome, we merely sketch the arguments here.

Lemma 3. (a) There is an L_c-formula ϕ_{lim} such that, for any precomputation F with $dom(F) = \tau$, we have $[\phi]^F_\tau = 1$ iff τ is a limit ordinal. Furthermore, the statement $\alpha = \beta + 1$ is expressable by an L_c-formulas $succ(\alpha, \beta)$.
(b) Let $A \subset \omega$ be finite. There is an L_c-formula $\phi_{liminf}(x, y)$ such that, for any $\tau \in On$, $a \in On$, $b \in A$ and any A-τ-precomputation F, $[\phi_{liminf}(a, b)]^F_\tau$ holds iff $b = \liminf((F(\iota))(a))_{\iota<\tau}$.
(c) Let P be an OTM-program, and let $\sigma = (i, \alpha, t)$ be a triple coding a state in the computation with P, where i is codes the current state of the program, α the head position and $t : \tau \to \{0, 1\}$ the tape content. There are L_c-formulas $\phi^P_{state}(i, \alpha, t, j)$, $\phi^P_{head}(i, \alpha, t, \beta)$ and $\phi^P_{tape}(i, \alpha, t, s)$ such, for any pre-computation F with $dom(F) = \gamma + 1$, $[\phi_{state}(i, \alpha, t, j)]^F_{\gamma+1} = 1$, $[\phi^P_{head}(i, \alpha, t, \beta)]^F_{\gamma+1} = 1$ and $[\phi^P_{tape}(i, \alpha, t, s)^F_{\gamma+1} = 1$ hold iff applying P in the state σ leads into the new state (j, β, t'), where $t' : \tau + 1 \to \{0, 1\}$ is given by $t'(\alpha) = s$ and $t'(\zeta) = t'(\zeta)$ for $\zeta \neq \alpha$.

Proof. (a) Take ϕ_{lim} to be $\forall x \exists y(x \leq y \wedge \neg(x = y))$. First assume that τ is a limit ordinal. Then $[\phi_{lim}]^F_\tau = 1 - [\exists x \forall y(\neg(x \leq y) \vee x = y))]^F_\tau$. Now $[\exists x \forall y(\neg(x \leq y) \vee x = y))]^F_\tau = 1$ iff there exists $x \in \tau$ with $[\forall y(\neg(x \leq y) \vee x = y)] = 1$, which is equivalent to $[\neg\exists y(x \leq y \wedge x \neq y)]^F_\tau = 1 \leftrightarrow [\exists y(x \leq y \wedge x \neq y)]^F_\tau = 0$, which means that there is no $y < \tau$ such that $[x \leq y \wedge x \neq y]^F_\tau = 1$, i.e. such that $x \leq y \wedge x \neq y$ holds. But such an x obviously cannot exist if τ is a limit ordinal. The other direction works in the same way, again by simply unfolding the definition of the truth predicate. The second statement is similarly immediate.

(b) As $A = \{a_1, ..., a_n\}$ is finite, we can define $\leq$ on A by taking $a < b$ to be $\bigvee_{a_i \leq b} a_i = a$. Now take $\phi(a, b)$ to be $\exists x \forall z(x \leq z \implies b \leq C(z, a) \wedge \forall x \exists z(x \leq z \wedge C(z, a) = b)$.

(c) The required formulas are immediate from P and the fact that limit ordinals are L_c-definable. To give an example, if P requires to change from state i to state j_1 when the symbol under the reading head (at position α) is currently ι_1 and to state j_2 when the symbol is ι_2, we can express this through the L_c-formula $\phi_i(\alpha, j) \equiv \exists\gamma(((\neg\exists\beta succ(\gamma, \beta) \wedge ((C(\gamma, \alpha) = \wedge j = j_1) \vee (C(\gamma, \alpha) = \wedge j = j_2)))$. □

Theorem 4. Let $f : On \to On$ be OTM-computable. Then f is IAM-computable.

Proof. Let P be an OTM-program for computing f. Suppose wlog that P uses $s \geq 3$ many states and put $A := \{0, 1, ..., s\}$. We will represent states of the OTM-computation as sequences $(a_i | i \in \alpha)$ where $a_0 \in \{1, 2, ..., s\}$ codes the inner state of the machine and the a_ι code the tape content. Let $b_i = a_{i+1}$ for $i \in \omega$ and $b_\iota = a_\iota$ otherwise. To express the head position, we put $b_\iota = 2$ if the ιth cell of the Turing tape contains a 0 and the head is currently at position ι, $b_\iota = 3$ if the ιth tape content is 1 and the head is currently at position ι; otherwise, the b_ι will just agree with the tape content.
Using lemma 3, one can now construct an L_c-formula ϕ such that $\mathbb{I}^\alpha_\phi$ represents the state and tape content of P at time α in the way we described. □

Theorem 5. Let $x \subset On$ be a set of ordinals. Then x is IAM-computable from a finite set of ordinals iff it is OTM-computable from a finite set of ordinals.

Proof. By [Koe], $x \subseteq On$ is OTM-computable from finitely many ordinal parameters iff $x \in L$. But it is not hard to see by adapting the theorem above that OTM-computations in finitely many parameters can be simulated by an IAM that hence every OTM-computable x is also IAM-computable. On the other hand, as IAM-computations are definable in L, every x IAM-computable from finitely many ordinal parameters must be an element of L. Hence the classes of IAM-computable sets of ordinals and OTM-computable sets of ordinals both coincide with the constructible sets of ordinals and hence with each other. □

Theorem 6. $f : On \to On$ is IAM-computable iff it is computable by an ordinal Turing machine (OTM) without parameters.

Proof. (Sketch) We saw above that an OTM can be simulated on an IAM.
For the other direction, we indicate how to simulate an IAM by an OTM. Let a finite A and an IAM-program ϕ be given.[11] To see how to emulate one computation step, assume we have safed the sequence $\mathbf{s} := (s_\iota | \iota < \tau)$ of IAM-states up to IAM-computing time τ so far on an extra tape T_1, separated by an extra symbol. The techniques from [OTM] for evaluating the bounded truth predicate can then be adapted to compute s_τ on a second tape, using a third tape as a scratch tape. For this, we compute, for each $\alpha \leq \tau$, $[\phi(\tau, \alpha, s)]^{\mathbf{s}}_\tau$ for each $s \in A$ until we find the unique $\bar{s}$ with $[\phi(\tau, \alpha, \bar{s})]^{\mathbf{s}}_\tau = 1$, so that $s_\tau(\alpha) = \bar{s}$. Finally, we copy s_τ to the end of T_1 to obtain a representation of $(s_\iota | \iota < \tau + 1)$. □

This shows, up to our analysis in section 3 and the restriction to working time and space On, that the intuitive concept of transfinite computability coincides with OTM-computability. Hence, we can finally close this section by stating our candidate for an $ICTT$:

Infinitary Church-Turing-Thesis: A function $f : On \to On$ is computable by the idealized agent of set theory following a deterministic rule iff it is computable by an OTM.

5. Conclusion and further Work

We have argued that there is an intuitive notion of transfinite computability and that rendering it precisely leads us to a notion of transfinite computability equivalent with ORM- and OTM-computability. Consequently, the constructible hierarchy was obtained as the realm of this idealized activity. This suggests that these models indeed capture some general intuitive concept and hence that results about these models can be interpreted as results about this notion. Accordingly, one should expect interesting applications to general mathematics: For example, one might consider measuring the complexity of an object or a function by the computational ressources necessary to compute it. This would give a precise meaning to the question whether certain objects granted to exist by indirect proofs can be 'concretely constructed', even if this construction is allowed to be transfinite. In particular, it suggests connections of transfinite computability to reverse mathematics as exhibited in [KoeWe].

However, our argument has the drawback of being model-dependent: We develop a certain notion of computability from the informal idea of an idealized agent, hopefully along plausible lines. It would be preferable to have a formal notion of

[11] Note that a variant of an OTM working with finitely many symbols $\sigma_1, ..., \sigma_n$ can be simulated by an OTM using only 0 and 1 by representing s_i as $\underbrace{0...0}_{n-i}\underbrace{1...1}_{i}$.

transfinite computation not refering to a particular model; this could be obtained by an appropriate axiomatization of transfinite computations similar to approaches that have been made in the classical case. (See e.g. [DeGu]. See also [KoeSy].)

Another question is whether a similar approach will work for other models like e.g. $ITTM$s. This is likely to be more difficult, as our coarse approach of approximating the activity of an idealized agent is not available here: As it is shown in [FrWe], there are natural alternative choices for the limit rules that lead to larger classes of computable functions.

References

[DeGu] N. Dershowitz, Y. Gurevich. A Natural Axiomatization of Computability and Proof of Church's Thesis. Bulletin of Symbolic Logic 14(3): 299-350 (2008)

[Fi] T. Fischbach. The Church-Turing-Thesis for Ordinal Computable Functions. Diploma Thesis. Bonn 2010.

[FrWe] S.-D. Friedman, P. Welch. Hypermachines. J. Symbolic Logic Volume 76, Issue 2 (2011), 620-636

[Ho] S. Hoffman. Kitcher, Ideal Agents and Fictionalism. Philosophia Mathematica (3) Vol. 12, pp. 3-17 (2004)

[Hog] M. L. Hogarth. Does General Relativity Allow an Observer to View an Eternity in a Finite Time? Foundations of Phyics Letters, Vol. 5, No 2, 1992

[ITTM] J.D. Hamkins and A. Lewis. Infinite Time Turing Machines. J. Symbolic Logic, 65(2), 567-604 (2000)

[Je] T. Jech. Set Theory. 3rd Millenium edition, revisited and expanded. Springer (2002)

[Ki] P. Kitcher. The Nature of Mathematical Knowledge. Oxford University Press (1983)

[Koe] P. Koepke. Ordinal computability. In Mathematical Theory and Computational Practice. K. Ambos-Spies et al, eds., Lecture Notes in Computer Science 5635 (2009), 280-289.

[KoeSy] P. Koepke, R. Siders. Minimality considerations for ordinal computers modeling constructibility. Theoretical Computer Science 394 (2008), 197-207

[KoeWe] P. Koepke, P. Welch. A Generalized Dynamical System, Infinite Time Register Machines, and $\Pi^1_1 - CA_0$. In CiE 2011. B. Löwe et al. (eds.), LNCS 6735, 152-159 (2011)

[Ma] Y. Matiyasevich. Hilbert's 10th problem. MIT Press, Cambridge, Massachusetts (1993)

[NeGe] P. Nemeti, G. Szekely. Existence of Faster than Light Signals Implies Hypercomputation already in Special Relativity. arXiv:1204.1773v1

[ORM] P. Koepke, R. Siders. Register computations on ordinals. Archive for Mathematical Logic 47 (2008), $529 - 548$.

[OTM] P. Koepke. Turing computations on ordinals. Bulletin of Symbolic Logic 11 (2005), $377 - 397$

[Sey] B. Seyfferth. Three models of ordinal computability. PhD thesis. Bonn 2012

[Tu] A. Turing. Intelligent Machinery. National Physical Laboratory Report. In Meltzer, B., Michie, D. (eds) 1969. Machine Intelligence 5. Edinburgh: Edinburgh University Press.

[Tu1] A. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society. 42, S. 230–265 (1937)

[Wa] H. Wang. From Mathematics to Philosophy. Routledge & Kegan Paul Ltd (1974)

The Bottleneck Selected-Internal Steiner Tree Problem: Hardness and Approximation

Yen Hung Chen

Department of Computer Science, Taipei Municipal University of Education,
Taipei 10042, Taiwan, R.O.C.
yhchen@tmue.edu.tw
http://tmue.edu.tw/~yhchen/

Abstract. Given a complete graph $G = (V, E)$, a positive length function on edges, and two subsets R of V and R' of R, the selected-internal Steiner tree is defined to be an acyclic subgraph of G spanning all vertices in R such that each vertex of R' does not belong to a leaf of the subgraph. The bottleneck selected-internal Steiner tree problem is to find a selected-internal Steiner tree T for R and R' in G such that the length of the largest edge in T is minimized. In this paper, we show the bottleneck selected-internal Steiner tree problem is NP-complete. We also show that there is no polynomial time approximation algorithm achieving a performance ratio of $(2 - \epsilon)$, $\epsilon > 0$, for the bottleneck selected-internal Steiner tree problem on metric graphs (i.e., a complete graph and the lengths of edges satisfy the triangle inequality) unless $P = NP$. Then, we extend to show that if the instance is not a metric graph (i.e., the lengths of edges do not satisfy the triangle inequality), there is no polynomial time approximation algorithm achieving a performance ratio of $(\alpha(|V|) - \epsilon)$, $\epsilon > 0$, for the bottleneck selected-internal Steiner tree problem unless $P = NP$, where $\alpha(|V|)$ is any computable function of $|V|$. Finally, we present the first known approximation algorithm with performance ratio of 3 for the bottleneck selected-internal Steiner tree problem on metric graphs.

Keywords: approximation algorithm, NP-complete, Steiner tree, terminal Steiner tree problem, selected-internal Steiner tree problem, bottleneck selected-internal Steiner tree problem, sensor network facility allocation, engineering change orders in VLSI design

1 Introduction

The *Steiner tree problem* (STP) is one of the fundamental problems in network design [7, 11, 12, 22]. Given an arbitrary graph $G = (V, E)$, a subset $R \subseteq V$ of vertices, and a positive length function on edges, a *Steiner tree* is used to find a connected and acyclic subgraph of G that spans all vertices in R. The given vertices R are usually referred to as *terminals* and other vertices $V \setminus R$ as *Steiner* vertices. The length of a Steiner tree is defined to be the sum of the lengths of all its edges. The STP is concerned with the determination of a Steiner tree with

minimum length in G [7, 11, 12, 22]. The STP was shown to be NP-complete [15] and MAX SNP-hard [2]. Hence, some polynomial-time approximation algorithms with constant ratios had been proposed [1, 3, 4, 17, 24, 30–33] instead of exact algorithms for the STP. The STP has many important applications in VLSI design, network communication, computational biology etc. [5, 7, 11, 12, 16, 22, 23, 25].

Motivated by the (sensor) network facility allocation and engineering change orders (ECO) in VLSI design, Hsieh and Yang [19] presented a variant of the STP, called the *selected-internal Steiner tree problem* (SISTP). Given a complete graph $G = (V, E)$, a positive length function on edges, and two subsets $R \subseteq V$ and $R' \subset R$, a Steiner tree for R in G is a selected-internal Steiner tree if all terminals in R' are internal vertices of this Steiner tree. The *selected-internal Steiner tree problem* (SISTP) is concerned with finding a selected-internal Steiner tree for R and R' in G whose total edge length is minimized [19, 26]. Without loss of generality, we assume $|R \setminus R'| \geq 2$ for the SISTP, otherwise the solution of SISTP may not exist. Then Hsieh and Yang [19] showed that the SISTP is NP-complete and MAX SNP-hard. They also presented a 2ρ-approximation algorithm for the SISTP on *metric graphs* (i.e., a complete graph and the lengths of edges satisfy the triangle inequality), where ρ is the best-known performance ratio for the STP (currently $\rho = \ln 4 + \epsilon \approx 1.39$ [4]). Li et al. [26] designed a $(\rho + 1)$-approximation algorithm which has the current best performance ratio for the SISTP on metric graphs. A similar problem to the SISTP is the internal Steiner tree problem. The *internal Steiner tree problem* (ISTP) is assumed that $R' = R$ and the purpose is to find a selected-internal Steiner tree for R in G with minimum total edge length. Huang et al. [21] gave a $(2\rho+1)$-approximation algorithm for the ISTP on metric graphs.

Although the SISTP is defined by a min-sum objective function, some applications of network and VLSI routing are considered in the bottleneck (min-max) objective function [7, 11, 12, 22]. Hence, we propose a variant of the SISTP with bottleneck objective function, called as the bottleneck selected-internal Steiner tree problem. Given a complete graph $G = (V, E)$, a positive length function on edges, and two subsets $R \subseteq V$ and $R' \subset R$, the bottleneck edge of a Steiner tree is an edge with the largest length in the Steiner tree. The *bottleneck selected-internal Steiner tree problem* (BSISTP) is concerned with the determination of a selected-internal Steiner tree for R and R' in G with minimum length of the bottleneck edge. For other related problems of the BSISTP, the bottleneck Steiner tree problem (BSTP) is to find a Steiner tree T in G with minimum length of the bottleneck edge [8, 13]. However, the bottleneck Steiner tree problem can be solved exactly in polynomial time [13]. A contrary problem of SISTP is the *partial terminal Steiner tree problem* (PTSTP) which is to find a minimum length Steiner tree T for R and R' in G such that all vertices of R' must be leaves of T [18, 20]. For the PTSTP on metric graphs, Hsieh et. al. [18, 20] showed that this problem is NP-complete and gave a $(2\rho - (\frac{\rho}{3\rho-2}))$-approximation algorithm. A special case of the PTSTP is the *terminal Steiner tree problem* (TSTP). The TSTP is assumed that $R' = R$ and the purpose is to find a minimum total

edge length Steiner tree for R in G such that all vertices of R must be leaves of T [3, 11, 22, 27, 28]. Since the TSTP had been shown to be NP-complete [27, 28], Chen, Lu and Tang [6], Fuchs [14], Drake and Hougardy [10], proposed 2ρ-approximation algorithms for the TSTP on metric graphs, independently, where ρ is the best-known performance ratio for the STP [4]. Then Martineza, Pinab, and Soares [29] designed a $(2\rho-(\frac{\rho}{3\rho-2}))$-approximation algorithm which has the current best performance ratio for the TSTP on metric graphs. Chen, Lu and Tang [6] also defined a bottleneck version of the TSTP, called as the bottleneck terminal Steiner tree problem (BTSTP). BTSTP is to find a Steiner tree T for R in G with minimum length of the bottleneck edge such that all vertices of R must be leaves of T. However, the BTSTP can be solved exactly in polynomial time [6]. In this paper, we show the BSISTP is NP-complete. We also show that there is no polynomial time approximation algorithm achieving a performance ratio of $(2-\epsilon)$, $\epsilon > 0$, for the BSISTP on metric graphs unless $P = NP$. Then, we extend to show that if the instance is not a metric graph (i.e., the lengths of edges do not satisfy the triangle inequality), there is no polynomial time approximation algorithm achieving a performance ratio of $(\alpha(|V|)-\epsilon)$, $\epsilon > 0$, for the BSISTP unless $P = NP$, where $\alpha(|V|)$ is any computable function of $|V|$. Finally, we present the first known approximation algorithm with performance ratio of 3 for the BSISTP on metric graphs.

The rest of this paper is organized as follows. In Section 2, we prove the NP-completeness and hardness of approximation results for the BSISTP. In Section 3, we describe our 3-approximation algorithm to solve the BSISTP. Finally, we give the concluding remarks in Section 4.

2 Hardness Results for the BSISTP

In this section, we show that the BSISTP is NP-complete even when the instance is a metric graph. This result also implies that the BSISTP cannot be approximated in polynomial time to within a ratio of $(2-\epsilon)$, $\epsilon > 0$, on metric graphs unless $P = NP$. Then it is easy to extend that unless $P = NP$, the BSISTP cannot be approximated in polynomial time to within a ratio of $(\alpha(|V|)-\epsilon)$, $\epsilon > 0$, for any computable function $\alpha(|V|)$ if the instance is not a metric graph. In order to show the BSISTP is NP-complete, we define the bottleneck selected-internal Steiner tree decision problem as follows.

Bottleneck Selected-Internal Steiner Tree Decision Problem
Instance: A positive integer m, a complete graph $G = (V, E)$, a positive length function ℓ on edges, and two subsets $R \subseteq V$ and $R' \subset R$.
Problem: Does there exist a selected-internal Steiner tree for R and R' in G such that the length of the bottleneck edge is less than or equal to m?

Now, we show that the bottleneck selected-internal Steiner tree decision problem is NP-complete by the *reduction* from the Hamiltonian path problem, which is a common NP-complete problem [9]. We define the Hamiltonian path problem as follows.

Hamiltonian Path Problem
Instance: A graph G and two vertices x and y.
Problem: Does there exist a path from x to y passing each vertex of G exactly once?

Theorem 1. *The decision version of the bottleneck selected-internal Steiner tree problem is an NP-complete problem even when the instance is a* metric graph *.*

Proof. First, it is easy to see that the bottleneck selected-internal Steiner tree decision problem is in NP. Then, we show the *reduction*: the transformation from the Hamiltonian path problem to the bottleneck selected-internal Steiner tree decision problem. Let a graph $G = (V, E)$ with two vertices v_x and v_y be an instance of the Hamiltonian path problem. Now, we construct an instance of bottleneck selected-internal Steiner tree decision problem, say a complete graph $\hat{G} = (\hat{V}, \hat{E})$, a positive length function ℓ on edges, two subsets $R \subseteq V$ and $R' \subset R$, and a positive integer m, as follows.

$m = 1$.
$\hat{V} = \{V\} \cup \{v_a\}$, where v_a is an auxiliary vertex.
$R = V$ and $R' = V \setminus \{v_x, v_y\}$.
For each edge $(u, v) \in \hat{E}$,

$$\ell(u, v) = \begin{cases} 1, \text{ if } (u, v) \in E \\ 2, \text{ otherwise.} \end{cases} \tag{1}$$

Clearly, the lengths of edges satisfy the triangle inequality. Now, we show that there is a Hamiltonian path of G from v_x to v_y if and only if there is a bottleneck selected-internal Steiner tree for R and R' in $\hat{G}$ such that the length of the bottleneck edge is m.

(Only if) Assume that there is a Hamiltonian path of G from v_x to v_y. It is clear this Hamiltonian path is a bottleneck selected-internal Steiner tree T for R and R' in $\hat{G}$ and the length of each edge of T is 1.

(If) there is a bottleneck selected-internal Steiner tree $T = (V_T, E_T)$ for R and R' in $\hat{G}$ such that the length of the bottleneck edge is 1. Clearly, all incident edges of v_a cannot be contained in E_T since the lengths of these edges are larger than 1. Hence, the vertex v_a does not belong to V_T. However, for all possible bottleneck selected-internal Steiner trees for R and R' in $\hat{G}$, only at most three vertices $\{v_x, v_y, v_a\}$ can be leaves. Hence, the bottleneck selected-internal Steiner tree T is a Hamiltonian path of G from v_x to v_y. □

The next theorems show hardness of approximation results for the BSISTP. For convenience, we let $len_b(T)$ denote the length of the bottleneck edge in any selected-internal Steiner tree T.

Theorem 2. *Assuming $P \neq NP$, there is no polynomial time approximation algorithm achieving a ratio of $(2 - \epsilon)$, $\epsilon > 0$, for the BSISTP on metric graphs.*

Proof. By the reduction of Theorem 1, let $\widetilde{T}$ denote the optimal solution in $\hat{G}$ for the BSISTP (i.e., a selected-internal Steiner tree for R and R' in $\hat{G}$ with minimum length of the bottleneck edge). By Theorem 1, the graph G has a Hamiltonian path iff $len_b(\widetilde{T}) = 1$ in $\hat{G}$ for the BSISTP. In other words, the graph G has no Hamiltonian path iff each selected-internal Steiner tree in $\hat{G}$ has length of the bottleneck edge larger than 1. Assume that there exists a $(2-\epsilon)$-approximation algorithm, denoted by $A_{2-\epsilon}$, for the BSISTP in polynomial time. Note that running the algorithm $A_{2-\epsilon}$ in $\hat{G}$ outputs all possible solutions of which the lengths of the bottleneck edges are either 1 or 2. Hence, if the $len_b(\widetilde{T}) = 1$ in $\hat{G}$ for the BSISTP, running Algorithm $A_{2-\epsilon}$ in $\hat{G}$ products a selected-internal Steiner tree T with the $len_b(T) = 1$. By Theorem 1, there exists a Hamiltonian path in graph G. If the $len_b(\widetilde{T}) = 2$ in $\hat{G}$ for the BSISTP, running Algorithm $A_{2-\epsilon}$ in $\hat{G}$ produces a selected-internal Steiner tree T with the $len_b(T) = 2$ that implies the graph G has no a Hamiltonian path. □

Theorem 3. *Assuming $P \neq NP$, there is no polynomial time approximation algorithm achieving a ratio of $(\alpha(|V|) - \epsilon)$, $\epsilon > 0$, for the BSISTP, where $\alpha(|V|)$ is any computable function of $|V|$.*

Proof. The Proof is similar to the Theorem 2. Let a graph $G = (V, E)$ with two vertices v_x and v_y be an instance of the Hamiltonian path problem. Now, we construct an instance of BSISTP, say a complete graph $\bar{G} = (\bar{V}, \bar{E})$, a positive length function ℓ on edges, and two subsets $R \subseteq V$ and $R' \subset R$.

$\bar{V} = \{V\} \cup \{v_a\}$, where v_a is an auxiliary vertex.
$R = V$ and $R' = V \setminus \{v_x, v_y\}$.
For each edge $(u, v) \in \bar{E}$,

$$\ell(u,v) = \begin{cases} 1, & \text{if } (u,v) \in E \\ \alpha(|V|), & \text{otherwise.} \end{cases} \tag{2}$$

Clearly, the lengths of edges do not satisfy the triangle inequality. We let $\widetilde{T}$ denote the optimal solution in $\bar{G}$ for the BSISTP. We also let $A_{\alpha(|V|)-\epsilon}$ be the $(\alpha(|V|) - \epsilon)$-approximation algorithm for the BSISTP in polynomial time. Clearly, the graph G has a Hamiltonian path iff $len_b(\widetilde{T}) = 1$ in $\bar{G}$ for the BSISTP by Theorem 1. In other words, the graph G has no Hamiltonian path iff each selected-internal Steiner tree in $\bar{G}$ has length of the bottleneck edge larger than 1. Note that running algorithm $A_{\alpha(|V|)-\epsilon}$ in $\bar{G}$ outputs all possible solutions of which the lengths of the bottleneck edges are either 1 or $\alpha(|V|)$. Hence, if the $len_b(\widetilde{T}) = 1$ in $\bar{G}$ for the BSISTP, running Algorithm $A_{\alpha(|V|)-\epsilon}$ in $\bar{G}$ produces a selected-internal Steiner tree T with the $len_b(T) = 1$. Hence, there exists a Hamiltonian path in graph G. If the $len_b(\widetilde{T}) = \alpha(|V|)$ in $\bar{G}$ for the BSISTP, running Algorithm $A_{\alpha(|V|)-\epsilon}$ in $\bar{G}$ products a selected-internal Steiner tree T with the $len_b(T) = \alpha(|V|)$ that implies the graph G has no a Hamiltonian path. □

3 A 3-Approximation Algorithm for the BSISTP

BSISTP (Bottleneck Selected-Internal Steiner Tree Problem)
Instance: A complete graph $G = (V, E)$ with a positive length function ℓ on edges, and two subsets $R \subseteq V$ and $R' \subset R$, where the length function ℓ is metric.
Problem: Find a selected-internal Steiner Tree for R and R' in G such that the length of the bottleneck edge is minimized.

In this section, we will present a 3-approximation algorithm to solve the above BSISTP, whose length function is metric, in polynomial time. For any arbitrary selected-internal Steiner tree T, we let $len_b(T)$ denote the length of the bottleneck edge in T. First, we use the exact algorithm for the BSTP [13] to construct a Steiner tree $S = (V_S, E_S)$ for R in G whose length of the bottleneck edge in S is minimized. Then, select any two vertices $\{v_a, v_b\}$ in $R \setminus R'$ (Note that $|R \setminus R'| \geq 2$) and recursively visit all vertices in V_S from v_a to v_b exactly once by taking a full walk of the tree S and skipping vertices, but without skipping more than three consecutive intermediate edges in S. For the Steiner tree S, we can easily convert S into a rooted tree with the root v_a. Hence, without loss of generality, let v_a be the root of S and let $P = (p_1 = v_a, p_2, p_3, \ldots, p_h = v_b)$ be the path between the two vertices v_a and v_b in S. Let T_r be a tree rooted at r, we use a recursive procedure $Traversal(T_r, flag)$ to determine the traversal order for a tree T_r and $flag \in \{0, 1\}$ is to determine the order of root visited. Now, we use Algorithm 1 to transform S into a selected-internal Steiner tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ (i.e., visiting each the vertex in V_S from v_a to v_b exactly once to form a Hamiltonian path in an induced subgraph $G' = (V_S, E')$ of G).

Algorithm 1: transforming S into a selected-internal Steiner tree $\mathcal{T}$
1. Let $V_{\mathcal{T}} = V_S$ and a global variable $\gamma = \phi$ be a vertex which is the last visited vertex before each recursive call the $Traversal(T_r, flag)$.
2. call Traversal(T_{p_1},1).
3. **For** $i = 2$ to h **do**
call Traversal(T_{p_i},0). **end For**

We describe the recursive procedure $Traversal(T_r, flag)$ as follows.

Procedure $Traversal(T_r, flag)$
1. **If** $flag = 1$ **then**
1.1 Visit the root r.
1.2 **If** the root r of T_r is not v_a **then** add an edge (γ,r) to $E_{\mathcal{T}}$. **endif**
1.3 Let $\gamma = r$.
endif
2. **For** each child β of the root except the vertex in P **do**
recursively call Traversal(T_β,1-flag). **end For**
3. **If** $flag = 0$ **then**
3.1 Visit the root r.
3.2 **If** the root r of T_r is not v_a **then** add an edge (γ,r) to $E_{\mathcal{T}}$. **endif**

3.3 Let $\gamma = r$.
endif

Lemma 1. *Algorithm 1 returns a selected-internal Steiner tree $\mathcal{T}$ for R and R' in G with $len_b(\mathcal{T}) \leq 3 * len_b(S)$.*

Proof. Clearly, after running Algorithm 1, $\mathcal{T}$ is a selected-internal Steiner tree for R and R' in G since the tree $\mathcal{T}$ only has two leaves $\{v_a, v_b\}$ in $R \setminus R'$. Notice that procedure Traversal visit the root frst if flag is 1, and last if flag is 0. For the procedure Traversal, we add an edge to $E_{\mathcal{T}}$ only in step 1.2 and 3.2. Without loss of generality, we visit each subtree of the tree T_{p_i} from left subtree to right subtree for $1 \leq i \leq h$ except the subtree $T_{p_{i+1}}$. For each edge (u, v) in $E_{\mathcal{T}}$, we let the vertex v be the next visited vertex after procedure Traversal visiting the vertex u. For each vertex u, we distinguish between the following two cases (i) flag=1(ii) flag=0. For the former case (i), Fig. 1 shows all possible cases of the next visited vertex v after visiting the vertex u according to the procedure Traversal. There are four cases of the vertex v: (1) v is a grandchild of u and $flag = 1$. (2) v is a sibling of u and $flag = 1$. (3) v is father of u and $flag = 0$ (4) v is a child (also, leaf of S) of u and $flag = 0$. For the case (1) and (2), it can easily be checked that the edge (u,v) in $E_{\mathcal{T}}$ is skipping 2 consecutive edges of the tree S. For the case (3) and (4), the the edge (u,v) in $E_{\mathcal{T}}$ is also an edge in E_S. For the later case (ii), Fig. 2 shows all possible cases of the next visited vertex v after visiting the vertex u according to the procedure Traversal. There are also four cases of the vertex v: (1) v is the child of sibling of u and $flag = 1$. (2) v is a sibling of father of u and $flag = 1$. (3) v is grandfather of u and $flag = 0$ (4) v is a sibling (also, leaf of S) of u and $flag = 0$. For the case (1) and (2), it can easily be checked that the edge (u,v) in $E_{\mathcal{T}}$ is skipping 3 consecutive edges of the tree S. For the case (3) and (4), it can easily be checked that the edge (u,v) in $E_{\mathcal{T}}$ is skipping 2 consecutive edges of the tree S. Then Fig. 3 shows the four cases of connecting the last visited vertex of tree T_{p_1} to the first visited vertex of tree T_{p_2} which is at most skipping 3 consecutive edges of the tree S . For $2 \leq i \leq h$, Fig. 4 shows the two cases of connecting the last visited vertex of tree T_{p_i} to the first visited vertex of tree $T_{p_{i+1}}$ which is at at most skipping 2 consecutive edges of the tree S. By triangle inequality, we have $len_b(\mathcal{T}) \leq 3 * len_b(S)$. □

Now, for clarification, we describe the approximation algorithm for BSISTP as follows.

Algorithm *APX*
Input: A complete graph $G = (V, E)$ with a positive length function ℓ on edges, and two sets $R \subset V$ of terminals and $R' \subset R$ of internal vertices, where the length function is metric.
Output: A selected-internal Steiner tree $\mathcal{T}$ for R and R' in G.
1. Use exact algorithm for the BSTP [13] to find a Steiner tree S in G.
2. If S is not a selected-internal Steiner tree for R and R' in G **then**
Use Algorithm 1 to transform S into a selected-internal Steiner tree $\mathcal{T}$.

The result of this section is summarized in the following theorem.

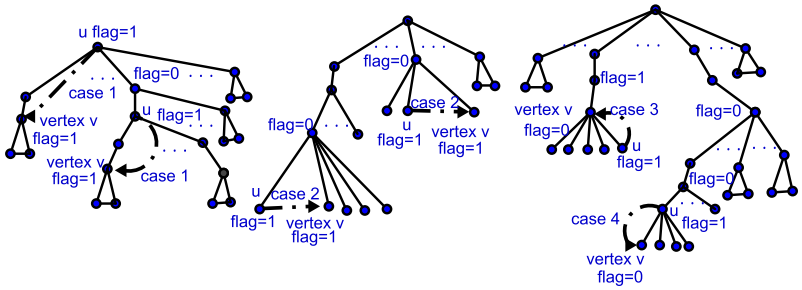


Fig. 1. Four cases of the edge (u, v) in $E_{\mathcal{T}}$ when $flag$ of vertex u is one, where the vertex v is the next visited vertex after visiting the vertex u. The bold dashed line in each case is an edge in $E_{\mathcal{T}}$.

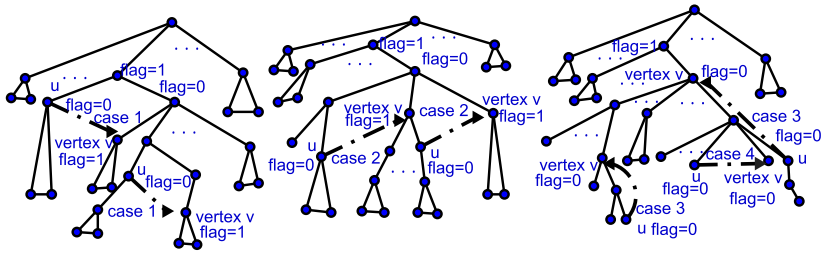


Fig. 2. Four cases of the edge (u, v) in $E_{\mathcal{T}}$ when $flag$ of vertex u is zero, where the vertex v is the next visited vertex after visiting the vertex u. The bold dashed line in each case is an edge in $E_{\mathcal{T}}$.

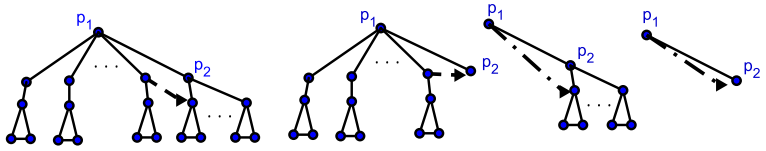


Fig. 3. Four cases of connecting the last visited vertex of tree T_{p_1} to the first visited vertex of tree T_{p_2}. The bold dashed line in each case is an edge in $E_{\mathcal{T}}$.

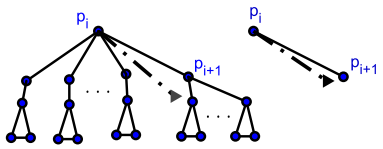


Fig. 4. Two cases of connecting the last visited vertex of tree T_{p_i} to the first visited vertex of tree $T_{p_{i+1}}$. The bold dashed line in each case is an edge in $E_{\mathcal{T}}$.

Theorem 4. *There is an algorithm finding a 3-approximation solution of the BSISTP of a metric graph in $O(|E|)$ time.*

Proof. Visiting each vertex in the Steiner tree S for Algorithm 1 takes $O(|V_S|)$ time. Hence, the time-complexity of Algorithm APX is dominated by the cost of the step 1 for running the exact algorithm for the BSTP (currently, the time complexity is $O(|E|)$ [13]). Let $\widetilde{T}$ and S be the optimal solutions for the BSISTP and BSTP for R and R' in G, respectively. Since $\widetilde{T}$ is also a Steiner tree for R in G, we have $len_b(S) \leq len_b(\widetilde{T})$. By Lemma 1, Algorithm APX returns a 3-approximation solution for the BSISTP. Therefore, the theorem is proved. □

4 Conclusion

In this paper, we have investigated the BSISTP and shown that the BSISTP is NP-complete on metric graphs. We have also shown that unless $P = NP$, the BSISTP cannot be approximated in polynomial time to within a ratio of $(2 - \epsilon)$ and $(\alpha(|V|) - \epsilon)$, $\epsilon > 0$, on metric and non-metric graphs, respectively. Finally, we have proposed the first known approximation algorithm with performance ratio of 3 for the BSISTP on metric graphs. For future research, improving the approximation ratio for the BSISTP is an immediate direction. Another direction for future research is whether the BSISTP is also NP-complete when the size of vertex sets R or R' is constant.

References

1. Berman, P., Ramaiyer, V.: Improved Approximations for the Steiner Tree Problem. Journal of Algorithms 17, 381–408 (1994)
2. Bern, M., Plassmann, P.: The Steiner Tree Problem with Edge Lengths 1 and 2. Information Processing Letters 32, 171–176 (1989)
3. Borchers, A., Du, D.Z.: The k-Steiner Ratio in Graphs. SIAM Journal on Computing 26, 857–869 (1997)
4. Byrka, J., Grandoni, F., Rothvos, T., Sanita, L.: An Improved LP-Based Approximation for Steiner Tree. In: Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC 2010), pp. 583–592. ACM, Cambridge (2010)
5. Caldwell, A., Kahng, A., Mantik, S., Markov, I., Zelikovsky, A.: On Wirelength Estimations for Row-Based Placement. In: Proceedings of the 1998 International Symposium on Physical Design (ISPD 1998), pp. 4–11. ACM, Monterey (1998)
6. Chen, Y.H., Lu, C.L., Tang, C.Y.: On the Full and Bottleneck Full Steiner Tree Problems. In: Warnow, T., Zhu, B. (eds.) COCOON 2003. Lecture Notes in Computer Science, vol. 2697, pp. 122–129. Springer Heidelberg (2003)
7. Cheng, X., Du, D.Z.: Steiner Tree in Industry. Kluwer Academic Publishers, Dordrecht, Netherlands (2001)
8. Chiang, C., Sarrafzadeh, M., Wong, C.K.: Global Router based on Steiner Min-max Trees. IEEE Transaction on Computer-Aided Design **9** 1318–1325 (1990)
9. Cormen, T.H., Leiserson, C.E., Rivest R.L., Stein, C.: Introduction to Algorithms. Third edition, MIT Press, Cambridge (2009)

10. Drake, D.E., Hougardy, S.: On Approximation Algorithms for the Terminal Steiner Tree Problem. Information Processing Letters 89, 15–18 (2004)
11. Du, D.Z., Smith, J.M., Rubinstein, J.H.: Advances in Steiner Tree. Kluwer Academic Publishers, Dordrecht, Netherlands (2000)
12. Du, D.Z., Hu, X.: Steiner Tree Problems in Computer Communication Networks. World Scientific Publishing Company (2008)
13. Duin, C.W., Volgenant, A.: The Partial Sum Criterion for Steiner Trees in Graphs and Shortest Paths. European Journal of Operations Research **97**, 172–182 (1997)
14. Fuchs, B.: A Note on the Terminal Steiner Tree Problem. Information Processing Letters 87, 219–220 (2003)
15. Garey, M.R., Graham, R.L., Johnson, D.S.: The Complexity of Computing Steiner Minimal Trees. SIAM Journal of Applied Mathematics 32, 835–859 (1997)
16. Graur, D., Li, W.H.: Fundamentals of Molecular Evolution (Second Edition). Sinauer Publishers, Sunderland, Massachusetts (2000)
17. Hougardy, S., Prommel, H.J.: A 1.598 Approximation Algorithm for the Steiner Problem in Graphs. In: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1999), pp. 448–453. ACM/SIGACT-SIAM, Baltimore (1999)
18. Hsieh, S.Y., Gao, H.M.: On the Partial Terminal Steiner Tree Problem. The Journal of Supercomputing 41, 41–52 (2007)
19. Hsieh, S.Y., Yang, S.C.: Approximating the Selected-Internal Steiner Tree. Theoretical Computer Science 381, 288–291 (2007)
20. Hsieh, S.Y., Pi, W.H.: On the Partial-Terminal Steiner Tree Problem. In: Proceedings of 9th International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN 2008), pp. 173–177. Sydney, NSW, Australia (2008)
21. Huang, C.W., Lee, C.W., Gao, H.M., Hsieh, S.Y.: The Internal Steiner Tree Problem: Hardness and Approximations. Journal of Complexity 29, 27–43 (2013)
22. Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem (Annuals of Discrete Mathematics 53). North-Holland: Elsevier, Amsterdam (1992)
23. Kahng, A.B., Robins, G.: On Optimal Interconnections for VLSI. Kluwer Academic Publishers, Boston (1995)
24. Karpinski, M., Zelikovsky, A.: New Approximation Algorithms for the Steiner Tree Problems. Journal of Combinatorial Optimization 1, 47–65 (1997)
25. Kim, J., Warnow, T.: Tutorial on Phylogenetic Tree Estimation. Manuscript, Department of Ecology and Evolutionary Biology, Yale University (1999)
26. Li, X., Zou, F., Huang, Y., Kim, D. Wu, W.: A Better Constant-Factor Approximation for Selected-Internal Steiner Minimum Tree. Algorithmica 56, 333–341 (2010)
27. Lin, G.H., Xue, G.L.: On the Terminal Steiner Tree Problem. Information Processing Letters 84, 103–107 (2002)
28. Lu, C.L., Tang, C.Y., Lee, R.C.T.: The Full Steiner Tree Problem. Theoretical Computer Science 306, 55–67 (2003)
29. Martineza, F.V., Pinab, J.C.D., Soares, J.: Algorithm for Terminal Steiner Trees. Theoretical Computer Science 389, 133–142 (2007)
30. Prommel, H.J., Steger, A.: A New Approximation Algorithm for the Steiner Tree Problem with Performance Ratio 5/3. Journal of Algorithms 36, 89–101 (2000)
31. Robins, G., Zelikovsky, A.: Tighter Bounds for Graph Steiner Tree Approximation. SIAM Journal on Discrete Mathematics 19, 122–134 (2005)
32. Zelikovsky, A.: An 11/6-Approximation Algorithm for the Network Steiner Problem. Algorithmica 9, 463–470 (1993)
33. Zelikovsky, A.: A Faster Approximation Algorithm for the Steiner Tree Problem in Graphs. Information Processing Letters 46, 79–83 (1993)

A History of Autonomous Agents: from Thinking Machines to Machines for Thinking

Stefania Costantini and Federico Gobbo[1]

DISIM – Dep. of Inf. Eng., Comp. Science, and Math.,
University of L'Aquila
via Vetoio, IT-67100 L'Aquila, Italy

Abstract. This paper analyses autonomous agents – one of the most important areas of Artificial Intelligence – under a historical perspective, in order to highlight how computational paradigms behave differently toward their conception, modeling and implementation. The main thesis is that autonomous agents, initially conceived as a useful metaphor to explore unexpected ways of the theory and practice of computation, eventually became an instrument to understand both the world of Nature and the virtual worlds created by humans through computing machines, which is particularly apt to describe the pervasive process of digitisation of information happening nowadays. To sustain this thesis, a special attention will be given to contemporary applications of multi-agent systems – such as gaming, swarm intelligence or social network simulation.

Keywords: Computational paradigms, Autonomous Agents, Multi-Agent Systems, History of Computing

1 Introduction

In this paper, we explore the world of autonomous agents, which represent one of the last frontiers of Artificial Intelligence (A.I.) and at the same time a long tradition of research in the field of applied computability, with unexpected results which could not be foreseen in the early days, in particular in its recent areas of application. In fact, the computing landscape drastically changed in the last decades: from a focus on standalone computer systems to a situation characterized by distributed, open and dynamic heterogeneous systems that must interact, and must operate effectively within rapidly changing circumstances and with increasing quantities of available information. In this context, agents constitute a suitable design metaphor, that provides designers and developers with a way of structuring an application around autonomous, communicative and flexible elements [28]. However, as observed by Nwana, there is a lack of agreement of what agents are, for at least two reasons:

> Firstly, agent researchers do not own this term in the same way as fuzzy logicians/AI researchers own the term fuzzy logic – it is one that is used widely in everyday parlance as in travel agents, estate agents, etc.

> Secondly, even within the software fraternity, the word agent is really an umbrella term for a heterogeneous body of research and development [30].

It is feasible to consider as the underlying assumption of the 'software fraternity' that 'agents' are pieces of software. The 'intelligence' of these agents is the capability to be *autonomous* in the environment, which can be summarised in this way: the ability to be reactive responding in a timely fashion to changes that occur around; the instinct to exhibit a goal-directed behaviour by taking the initiative; the wish to interact with other entities through a common language, the skill of choosing a plan to reach one or more goals, preferably through the learning from past experience. In the sequel, we will use the expression Autonomous Agents (A.A.) to indicate this particular family of softwares, while the term 'agent' alone would indicate both entities based on biology (e.g. human beings, other animals) or Turing machines (i.e., robots in the physical world; softbots in wide computer networks; taskbots embedded in specific hardware, etc.). Here, we follow Wooldridge and Jennings, who define an agent – valid in general – as 'one who, or which, exerts power or produces an effect' [38], while this effect is exercised through actions in a given environment, that can influence the behaviour of an agent, which is dynamically responsive. However, they go along with the definition of software-based agents where the first property is rightly *autonomy*, i.e., the ability to operate without the direct intervention of humans or others, while having some kind of control over their own actions and internal state. Moreover, A.A. usually show reactivity (to the environment, and this implies some kind of perception), proactivity (opportunistic, goal-directed behaviour, taking the initiative where appropriate) and social abilities (i.e., appropriate interaction with other agents).

The paper is structured as follows. In section 2, the birth of A.A. will be recalled as a novel programming paradigm, and how this paradigm was interpreted by the different communities of computer scientists, according to the schools they belong. Then, section 3 will explore what happens when autonomous agents are put together, forming coherent Multi-Agent Systems (MAS). Finally, the concrete applications of MAS will be discussed in section 4, in order to show how the main areas of application of MAS influenced backwards the modelling of MAS themselves and eventually the very concept of A.A. In the conclusion (section 5), an evaluation of A.A. as a paradigm of computation apt to interpret the current trends of the computational turn, i.e., after year 2000.

2 Autonomous Agents as a Programming Paradigm

Paraphrasing Roland Barthes, the Agenthood Degree Zero is the seminal paper by Shoham published in 1990, where A.A. were proposed as a specific programming paradigm [36]. It is worth noticing, that in that year we were in the middle of the so-called 'second winter' of A.I.: according to Russell and Norvig, that period started in 1988, after the fall of the A.I. industry and the failure of the

'Fifth Generation' project [35]. Already in 1984, McCarthy – the inventor of the expression A.I. itself in 1956, and since then a leading figure in the community – had expressed dissatisfaction for the results of expert systems of the time, advocating for a movement back to the original goals – i.e., to general, flexible, autonomous systems – on a new basis [29]. From the other side, in 1990 Brooks launched his program of a *nouvelle A.I.*, no more purely based on software, and hence disincarnate, but rather embodied in robots [5].

The novelty brought by Shoham is simple: instead of programming objects, which pass messages in response to methods, in principle without constraints in terms of parameters (state of basic unit) or types (messages), it is more effective to constrain the object as an A.A., with constraints on methods, for instance 'honesty, consistency' and on parameters, like 'beliefs, commitments, capabilities, choices' [36]. Therefore, agent oriented programming is proposed as a specialisation of object oriented programming. The success of A.A. was great and rapid: only five years later, Russell and Norvig will publish the first edition of the most important textbook of A.I. ever in 1995, where they state explicitely and undoubtedly that 'A.I. is the study of agents' [35].

However, a relevant part of the A.I. community of that period (but not only) was not keen to think – and hence program – in terms of object-orientation, whose relation with A.I. passes through the concept of Marvin Minsky's frame, but it is extraneous to the symbolic approaches of A.I. (see [6] for details on the history of the object-oriented paradigm). According to Wooldridge and Jennings, it is not by chance that soon the instruments of Computational Logic (CL) were advocated to model agenthood, starting from Daniel Dennet's concept of *intentional system*: according to him, the behaviour of an agent (and an A.A. in particular) can be predicted by the method of attributing belief, desires and rational acumen [38]. In particular, in 1991 Rao and Georgeff proposed an abstract architecture called BDI (Belief, Desire, Intention) in order to face the problem of continuous and non deterministic evolution of the environment [32]. BDI is formally define in terms of modal logic, where Belief, Desire, Intention are understood as modalities. In the first place, it is necessary that a component in the system state represent the world information and update appropriately this information after each sensing action. This component, called *belief*, may be implemented as a variable, a database, a set of logical expressions or some other data structure. Beliefs are essential because the world is dynamic (past events need therefore to be remembered), and the system only has a local view of the world. In A.I. terms, beliefs represent knowledge of the world. However, in computational terms, beliefs are just some way of representing the world state, be it the value of a variable, a relational database, or symbolic expressions in predicate calculus.

Moreover, as the system is resource bounded, it is desirable to cache important information rather than recompute it from base perceptual data. But an agent must have information also about the objectives to be accomplished. Desires – within the BDI architecture, or, more commonly though somewhat loosely, "goals" – form another essential component of system state. The rep-

resentation of desires implies the modellisation of the information about the objectives to be accomplished, the priorities and payoffs associated with the various objectives. Thus, desires can be thought as representing the "motivational" component of an agent. In order to define the process of choice of actual agent's course of actions, the definition of the *selection* function is crucial, which determines which desires will be actually pursued. The selection function involves means-end analysis, and the weighting of competing alternatives. Also, in a changing environment the possibility of reconsideration in presence of new circumstances must be foreseen. The selection function generates *intentions*, which are desires that the agent has committed to achieve. Thus, intentions capture the "deliberative" component of an agent. Intentions are stronger than desires:

> My desire to play basketball this afternoon is merely a potential influencer of my conduct this afternoon. It must vie with my other relevant desires [...] before it is settled what I will do. In contrast, once I intend to play basketball this afternoon, the matter is settled: I normally need not continue to weigh the pros and cons. When the afternoon arrives, I will normally just proceed to execute my intention. [4]

Formally, an intention is a feasible desire, i.e., a desire which can be satisfied in practice, as the agent is able to devise and execute a plan to achieve it. Computationally, intentions may simply be a set of executing threads that can be appropriately interrupted upon receiving feedback from the possibly changing world. Rather than try to recreate every new plan from first principles, practical BDI architecture [33] includes a support structure to manage the planning process of an agent. This structure that works as a cache, contains parameterized "plans" for use in future situations. Semantically, these plans can be viewed as a special kind of belief. It is important to notice that, according to Bratman [4], agents should persist with their intentions (as long as they are satisfiable), i.e., they should be committed to those intentions. Commitment is an important property that, if implemented, creates a safer scenario for the other agents, including the user: in fact, they are able to expect that an agent will not arbitrarily modify its behaviour, as discussed also by Cohen and Levesque [8], that provided one of the best-known and most influential contributions to the study of intentions and commitments and in general to the theory of "rational agents".

The BDI approach has been realized (though in a simplified form) in the AgentSpeak programming language [33, 34], that has several implementations and extensions. Among the existing BDI virtual machines, we shall mention at least Jason[1] and JACK[2]. However, Kowalski pointed out that BDI agent-based systems, though formally defined in terms of modal logics, only have implementations (like those mentioned above) defined in procedural languages, which are based on the mathematical model of computation originally proposed by Emil Post; such implementations are normally considered out of the computational

[1] See the official web page for details: `http://jason.sourceforge.net/wp/`.

[2] See the official web page for details `http://www.agent-software.com.au/products/jack`.

Logic tradition [25] – however, Martin Davis has argued that the uniform algorithm by Robinson on which is based for example the Prolog programming language is nothing more than the L.C.M. process by Post, hence there is a small point of contact between the two traditions (see [15] for details).

The Abductive Logic Programming (ALP) agent model defined by Kowalski and Sadri in 1999 uses logic both at a level of specification and implementation [26], overcoming the limits of the implementations of BDI. The ALP agent model shows how logic can support every capability of A.A., maintaining both a close connection with the environment and the power of reasoning offered by a symbolic approach. However, abductive reasoning should be extended by other known techniques in CL, i.e., temporal reasoning, constraint logic programming, and preference reasoning based on logic programming with priorities. Therefore, the KGP (Knowledge, Goals and Plans) agent model [23, 2] was proposed. The formulation of the KGP architecture, even if takes BDI model as a starting point, uses CL in an innovative manner in order to specify the individual state of an agent, its reasoning capabilities, state transitions and control. Through these capabilities, KGP agents are able to live in open and dynamic environments. The authors emphasize that CL allows an agent to maintain a view of the environment, decide what its goals should be depending on the current circumstances, plan (incrementally) for these chosen goals and interleave this with plan incremental execution, react to environment received information, re-evaluate previous decisions in the light of the new information and adapt as necessary by changing or augmenting its goals and plan. For a full treatment of the CL foundations of KGP agents, see [24].

Many other approaches to defining agent architectures in computational logic exist, among which one has to mention at least MetateM [16] (based on modal temporal logic), 3APL [22] (which facilitates specification of cognitive agent behaviour using actions, beliefs, goals, plans, and rules), Impact [37] (which provides techniques and tools to build agents, also on top of existing legacy code), and DALI [9, 11, 12] (a prolog-like logic programming language, where the reactive, proactive and social behaviour of DALI agents is triggered by several kinds of events: external, internal, present and past events). For a recent survey on computational logic agents, the reader may refer to [17].

Despite the differences among the different approaches, a logical agent is in general based upon an "agent program" which consists of a knowledge base and of a set of rules aimed at providing the entity with the needed capabilities. Rules may include object-level rules and meta-(meta-...)rules that determine the agent behaviour. The knowledge base may itself include rules, which either define knowledge (and meta-knowledge) in an abstract way or constitute part of the agent knowledge. The knowledge base constitutes in fact the agent "memory" while rules define the agent behaviour. An underlying inference engine, or more generally a control mechanism, puts an agent at work. Agents evolve in time, as a result of both their interaction with the environment and their own self-modifications. In summary, all logical agent-oriented architectures and languages, or "agent models", exhibit at least the following basic features (for

a general discussion about logical agent models the reader may see, e.g., [17] and [14], and for a general logical semantics for evolving agents [13]): a *logical core*, that for instance in both KGP and DALI is a logic program; *reactivity*, i.e., the capability of managing external stimuli; *proactivity*, i.e., the capability of devising, managing and pursuing internal "initiatives"; the capability of performing actions and executing plans; the capability of recording and elaborating "experiences" related to what has happened and has been done in the past; the capability of communicating with other agents; a basic cycle that interleaves the application of formerly specified capabilities; e.g., in DALI the basic cycle is integrated within the logical core via an extended resolution, while in KGP the basic cycle has a meta-level definition; in both cases, this cycle can be customized by the programmer under various respects.

The above features allow logical agents to be *intelligent*, so as to face changing situations by modifying their behaviour, or their goals, or the way to achieve their goals. This in fact requires agents to be able to perform, interleave and combine various forms of commonsense reasoning, possibly based upon different kinds of representation.

3 Autonomous Agents put together

Agents are social entities and this is a relevant property: they exist in an environment containing other agents, with which they will generally be expected to interact in some way. In fact, application domains are so often large, sophisticated and unpredictable that the only solution is to develop a considerable number of specialized entities, each one with a reduced point of view of the world. In 2001, Adams observes that not only a MAS is more complex than a single agent system, but further issues should be considered, such as the 'communication mechanisms, environmental and world knowledge maintenance as well as communication, and societal issues such as which agent is assigned to a particular task' [1]. In a MAS there is no global system control, data are decentralized and computation is asynchronous. As observed by Wooldridge and Jennings, the classic "divide and conquer" methodology of traditional software engineering is not apt for a MAS, as 'agents must be able to interact in a flexible, context dependent manner (hence the need for social abilities, responsiveness, and proactiveness) rather than through some fixed and predetermined set of interface functions. Also, the unpredictability of the domain means that the agents must be both responsive to change and proactive.' [38] Moreover, interaction is one of the most important features. When multiple entities share an environment, it is not trivial to control their behaviours and to synchronize their activities: individual needs can determine overall system goals failure. In order to solve common problems coherently, agents must communicate amongst themselves and coordinate their activities. In fact, no agent possesses a global view of the entire agency to which it belongs, as this is simply not feasible in any community of reasonable complexity. Consequently, agents have only local views, goals and knowledge that may interfere with rather than support other

agents' actions. Coordination is vital to prevent chaos during conflicts. Generally, communities of agents are used to solve a problem collectively, but sometimes agents have egoistic interests that pursue the detriment of others. These two general strategies have generated two modalities of interaction: cooperation and competition. The basic premise of this coordination form is that an agent cannot solve an assigned problem using local resources/expertise. Thus it will decompose the problem into sub-problems and try to find other willing agents with the necessary resources/expertise to solve these sub-problems. The assignment of the sub-problems is solved by a contracting mechanism. It consists of a manager agent contract announcement, submission of contracting agents bids in response to the announcement, and the evaluation of the manager submitted bids, which leads to awarding a sub-problem contract to the contractor(s) with the most appropriate bid(s). The study of how groups of agents work is a central issue in MAS, and results from game theory are applied accordingly to cope with it: e.g., multi-modal logics were recently proposed to formalise teamwork in MAS [10].

The comparison with real-world, human societies is immediate and striking: the natural consequence is to consider MAS as a reasonable – although rather simplified – model of Nature and human societies.

4 Multi-Agent Systems in action

Complex real-world problems require to cope with complexity, meaning both at a level of computability and of domain modelling. The joint behaviour of agents – dealing with cooperation and competition – has proven to be a good way to cope with complexity in a lot of areas in recent years. In 2005, a survey of the state-of-the-art on cooperative MAS by Panait and Luke states that:

> Much of the multi-agent learning literature has sprung from historically somewhat separate communities – notably reinforcement learning and dynamic programming, robotics, evolutionary comutation, and complex system... We think that there are two features of multi-agent learning which merit its study as a field separate from ordinary machine learning. First, because multi-agent learning deals with problem domains involving multiple agents, the search space involved can be unusually large [...] Second, multi-agent learning *may* involve *multiple learners*, each learning and adapting in the context of others. [31]

The interesting fact is, that MAS are more and more considered a field of research *per se*, with its own unique features and open problems – such as the machine learning issue. Another important trend is given by the interaction between A.A. and biological agents in hybrid environments. On one hand, the decrease in the costs of robots at the end of the 20^{th} century let researchers to apply MAS in real-world environments: one of the most popular examples is the RoboCup competition, founded in 1997, where robots play football matches.[3] On

[3] For details, see the official web page: `http://www.robocup.org/`, retrived in January the 13^{th}, 2013.

the other hand, 'serious gaming' – a field of research where computer games are used for serious purposes, i.e., besides entertainment, typically education, but also economic simulation – permits to observe A. A. play with human agents in the same virtual environment. Hybrid environments (both in the real or virtual worlds) raise again the A.I. issue of emotion, applied to A.A. For instance, Bates describe A.A. not only in terms of reactivity and goals, but also in terms of emotions and social behaviour [3]. In this emotional architecture, happiness and sadness occur when the agent's goals succeed or fail. Pride, shame, reproach and admiration arise when an action is either approved or not. These judgments are made according to the agent's standards, which represent moral beliefs and personal standards of performance. Foner proceeds beyond Bates, in describing what an agent must be by studying a software agent named Julia, which interacts with people in MUDs (Multi-User Dungeons), a hybrid environment where people chat and get together with A.A. in rooms, originally text-based, then with avatars – the most famous being Second Life, see [27] for a critique. This case study includes issues of personality, anthropomorphism, expectations and other human attributes [18].

5 Conclusion

Despite the apocalyptic alarms of the intelligent machines conquering our real world and so putting humanity in enslavement, facts show a different reality: nowadays, humans more and more interact through the 'internet galaxy' (Castells) eventually reshaping their social (and political), at least since year 2000 [7]. According to Floridi, the digitisation of the infosphere (i.e., the global complex system made of processes, services and relations between informational organisms) was made in three steps: first, formal and natural language data, then multimedia documents (sounds, images and motions), became digitised; second, user interfaces passed from punched cards to visualisation (WIMP systems, i.e., Windows, Icon, Manu, Pointer) and then manipulation (3D graphic interfaces, immersive environment with avatars); third, convergence and integration of the digitised data through the connection of environment via computer networks [20]. The result is that the underlying concept that unifies every type of agents is *information*: informational organisms (inforgs) are defined as biological agents acting in a environment through engineered artefacts, producing information [19]. After the digitisation of the infosphere, a new class of *computational* inforgs is become more and more important, where at least part of the engineered artefacts are based on a Von Neumann machine, with specific traits [21]. According to this line, A. A. can be seen as a special case of computational inforgs, where the engineered artifact shows a degree of autonomy in dealing with the human counterpart. Therefore, MAS can be an exceptional promising field of research in order to explain and shape the hybrid environments which constitute the infosphere in which agents of all kind live together.

References

1. Adams, J. A.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence: A Review. AI Magazine, 22(2).
2. Bracciali, A., Demetriou, N., Endriss, U., Kakas, A., Lu, W., Mancarella, P., Sadri, F., Stathis, K., Terreni, G., Toni, F.: The KGP model of agency: Computational model and prototype implementation. In: Global Computing: IST/FET International Workshop, Revised Selected Papers. LNAI 3267, 340–367. Springer-Verlag, Berlin (2005)
3. Bates, J.: The role of emotion in believable agents. Commun. of the ACM, 37(7), 122–125 (1994)
4. Bratman, M. E.: What is intention?. In Cohen, P. R., Morgan, J. L., and Pollack, M. E. (editors), Intentions in Communication, pages 15-32. The MIT Press: Cambridge, MA (1990).
5. Brooks, R. A.: Elephants don't play chess. Robot. Auton. Syst, 6(1–2), 3–15 (1990)
6. Capretz, L. F.: A brief history of the object-oriented approach. SIGSOFT Softw. Eng. Notes, 28(2), 1–10 (2003)
7. Castells, M.: The Internet Galaxy. Oxford University Press (2001)
8. Cohen, P. R. and Levesque, H. J.: Intention is choice with commitment. Artificial Intelligence, 42:213-261 (1990).
9. Costantini, S.: Towards Active Logic Programming. In: A. Brogi and P.M. Hill (editors), Proc. of 2nd International Works. on Component-based Software Development in Computational Logic (COCL99), PLI99, Indexed by CiteSeerX (1999).
10. Dunin-Kçplicz, B. and Verbrugge, R. A logical view on teamwork. In: J. van Eijck and R. Verbrugge (eds.), Games, Actions and Social Software: Multidisciplinary Aspects, Texts in Logic and Games, FoLLI subseries of Springer Lecture Notes in Computer Science, volume 7010, Springer Verlag, Berlin, 2012, pp. 184-212.
11. Costantini, S., Tocchio, A.: A logic programming language for multi-agent systems. In: Logics in Artificial Intelligence, Proc. of the 8th Europ. Conf.,JELIA 2002. LNAI 2424, Springer-Verlag, Berlin (2002)
12. Costantini, S., Tocchio, A.: The DALI logic programming agent-oriented language. In: Logics in Artificial Intelligence, Proc. of the 9th European Conference, Jelia 2004. LNAI 3229, Springer-Verlag, Berlin (2004)
13. Costantini, S., Tocchio, A.: About declarative semantics of logic-based agent languages. In Baldoni, M., Torroni, P. (editors), Declarative Agent Languages and Technologies. LNAI 3904, 106–123, Springer (2006).
14. Costantini, S., Tocchio, A., Toni, F., Tsintza, P.: A multi-layered general agent model. In: AI*IA 2007: Artificial Intelligence and Human-Oriented Computing, 10th Congress of the Italian Association for Artificial Intelligence. LNCS 4733, Springer-Verlag, Berlin (2007).
15. De Mol, L.: Formalism. The success(es) of a failure. In: Moktefi, A., Moretti, A., Schang, F. (eds). Let's be logical, College publications, (to appear)
16. Fisher, M.: Metatem: The story so far. In: Bordini, R.H., Dastani, M., Dix, J., Fallah-Seghrouchni, A.E. (editors), PROMAS. Lecture Notes in Computer Science 3862 of , 3–22, Springer (2005).
17. Fisher, M., Bordini, R.H., Hirsch, B., Torroni, P.: Computational logics and agents: a road map of current technologies and future trends. Computational Intelligence Journal 23(1), 61–91 (2007)
18. Foner, L.: Entertaining Agents: A Sociological Case Study. Proc. of the 1st International Conference on Autonomous Agents. In: Mauldin, Chatterbots, Tiny-muds, and the Turing Test. In: Moktefi, A., Moretti, A., Schang, F. (eds).

19. Floridi, L.: The Philosophy of Information. Oxford University Press (2011)
20. Floridi, L.: Philosophy and Computing. Oxford University Press (1999)
21. Gobbo, F., Benini, M.: The Minimal Level of Abstraction in the History of Modern Computing. Philos. Technol., special issue (2013)
22. Hindriks, K.V., de Boer, F., van der Hoek, W., Meyer, J.C.: Agent programming in 3APL. Autonomous Agents and Multi-Agent Systems 2(4) (1999)
23. Kakas, A.C., Mancarella, P., Sadri, F., Stathis, K., Toni, F.: The KGP model of agency. In: Proc. ECAI-2004. (2004)
24. Kakas, A., Mancarella, P., Sadri, F., Stathis, K., Toni, F.: Computational Logic Foundations of KGP Agents. Journal of Artificial Intelligence Research, 33, 285–348 (2008)
25. Kowalski, R.: Reasoning with Conditionals in Artificial Intelligence. Draft. Revision of December 2009, (2009)
26. Kowalski, R., Sadri, F.: From Logic Programming towards Multi-agent Systems. Annals of Mathematics and Artificial Intelligence, 25, 391–419 (1999)
27. Lanzarone, G. A.: In Search of a New Agenda. APA Newsletter Computing and Philosophy, 7(1), 18–21 (2007)
28. Luck, M., McBurney, P., Preist, C.: A manifesto for agent technology: Towards next generation computing. Autonomous Agents and Multi-Agent Sytems 9, 203–252 (2004)
29. McCarthy, J.: Some expert systems need common-sense. In: Pagels, H. (ed). Proc. of a symposium on Computer Culture: The Scientific, Intellectual and Social Impact of the Computer. Annals of the New York Academy of Sciences (1984)
30. Nwana, H., S.: Software Agents: An Overview. Knowledge Eng. Review, 11(3), 1–40 (1996)
31. Panait, L., Luke, S.: Cooperative Multi-Agent Learning: The State of the Art. Autonomous Agents and Multi-Agent Systems, 11(3), 387–434.
32. Rao, A. S., Georgeff, M.: Modeling rational agents within a BDI-architecture. In: Allen, J., Fikes, R., Sandewall, E. (eds). Proc. of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91), 473–484 (1991).
33. Rao, A. S. & Georgeff, M.: BDI agents: from theory to practice. In: Victor R. Lesser and Les Gasser (editors), Proceedings of the First International Conference on Multiagent Systems, 312–319, The MIT Press, (1995).
34. Rao, A. S.: *AgentSpeak(L)*: BDI agents speak out in a logical computable language. In: Walter Van de Velde and John W. Perram (editors), Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Lecture Notes in Computer Science 1038, 42–55, Springer (1996).
35. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentiche-Hall (1995)
36. Shoham, Y.: Agent Oriented Programming. Technical Report STAN-CS-90-1335, Computer Science Department, Stanford University (1990)
37. Subrahmanian, V.S., Bonatti, P., Dix, J., Eiter, T., Kraus, S., Ozcan, F., Ross, R.: Heterogeneous Agent Systems. MIT Press/AAAI Press, Cambridge, MA, USA (2000)
38. Wooldridge, M. J., Jennings, N. R.: Agent Theories, Architectures, and Languages: a Survey. In: Wooldridge, M. J., Jennings, N. R. Intelligent Agents. Lecture Notes in Computer Science, Volume 890, Berlin: Springer-Verlag. 1–39 (1995)

Brute Force is not Ignorance

Joseph Davidson and Greg Michaelson

School of Mathematical and Computer Sciences
Heriot-Watt University, Edinburgh, Scotland
{jrd5/G.Michaelson}@hw.ac.uk

Abstract. Chaitin's notion of program elegance, that is of the smallest program to satisfy some specification, does not explicitly take account of the balance between a formal notation's expressive power and the richness of its semantics. To explore this wider space of elegance, we are investigating realisations of the Busy Beaver game (BBG) which involves finding programs of given size that produce maximal outputs.
Like elegance, BBG is undecidable. Canonically, BBG is represented using Turing machines, but there has been very little investigation into alternative formulations. Thus, we are exploring BBG in a number of classic models of computability, using empirical and analytic heuristics to find optimal BBG instances. Such experiments will be used as a basis for drawing comparisons between the expressive power of Turing machines and other models of computation with a view to gaining a deeper understanding of the expressive power of computer languages.
In this paper, we re-introduce the Random Access Stored Program machine (RASP) and build an analogue of the BBG for these machines. Though the BBG for any fixed precision RASP model is trivially computable, the expressivity of the model renders exhaustive search infeasible as we increase the size of our machine. Thus, we explore the space of BBG RASP machines using both brute force and genetic search methods.

Keywords: Elegance, Busy Beaver, Computability, Genetic Algorithms, Brute Force, RASP Machine

1 Motivation

Chaitin has extensively investigated the concept of elegant programs [3]. A program P, calculating a function f is said to be elegant if there is no other program which both calculates f and is smaller (by source code characters) than P.

Using Lisp, Chaitin has produced a contradictory program $elegantFinder$ which is combined with the rules to some formal axiomatic system (FAS) FAS. Say we wish to find the shortest expression F for a function f. We invoke $elegantFinder$ and FAS such that they enumerate all programs in size order and test them each to see if they calculate f and are therefore F.

Once $elegantFinder$ has found F, it runs F in order to obtain its value and returns it. In doing this, we can see that $elegantFinder$ also calculates f.

This becomes paradoxical when we consider the relative sizes of the programs involved. If F is smaller (in number of characters) than $|elegantFinder|+|FAS|$, then our elegant program is F. However, if F is larger (possibly calculating a very complex f), then because $elegantFinder$ runs F once it has been found, it turns out that $elegantFinder$ is actually our elegant program for f!

The problem of finding a our elegant program F is quite convincingly uncomputable, however it also raises some related questions about the expressive power of languages. Taking the size of interpreters/compilers into account, is a function in a high-level language like Lisp any smaller than in a lower level language like C, or even an Assembly-like language? Intuitively, we can see that lower level languages or models require simpler 'back end' infrastructure to be executed by the commonplace Von Neumann architecture, but is there a sweet spot in the ratio of program complexity to interpreter complexity where we can develop short programs and have them execute/compile quickly? And if there is, can we deduce or prove where that spot is and develop an 'elegant language' to take advantage?

Elegance is undecidable. However, we hypothesise that by realising the same computations in different models of computation, we can obtain heuristic information that allows us to draw useful comparisons of elegance. Thus, this paper discusses the implementation of the Busy Beaver game (BBG) in the Random Access Stored Program (RASP) architecture – which is similar to the RAM machine – and considers the size of the resulting search space and attempts to navigate it. The BBG itself is a problem in which every solution is elegant (we can phrase it as looking for the smallest machine to output a given integer), so producing this problem in both representations will help give us the experimental data that we need.

2 The Busy Beaver Game

The 'Busy beaver game', was formulated by Radó [8] in order to show an example of a simple undecidable problem. It is the problem of defining an instance of a Turing machine such that, when started on a blank tape it produces as much output as possible before halting. The Turing machine is the canonical model for exploring the busy beavers and it is what the majority of the hunters use due to its well defined operations and the relative simplicity of instance generation. When we limit our machines to a particular size of states – and symbols when Brady extended his machine in 1988 [1] – we create a competition between machine designers to find the 'champion machine' for that size. The 'champion machine' for a class is defined as the machine which either produces the most output or performs the highest number of shifts before halting, compared to other machines of that class.

Radó formalises his game using n 'state' Turing machines and uses this as a method of classifying the machines. He defines a valid entry into the BB-n classification as a pair (M, s) where M is the Turing machine of n states and s is the exact number of steps that M has run for before halting.

There are two competitions in the BBG: The greatest number of shifts, and the greatest number of non blank symbols left on the tape. We characterise the shift number for a given machine as s, and the maximum shifts for a class as S. Likewise, the non-blank symbols for an individual machine is σ and for a class, we use Σ.

This method gives us an easy way to validate and score an entry, we simply run M for s steps and – if the machine has halted – record the number of non blank symbols on the tape. If the output is larger than the previous record, we call M our new champion.

After Brady extended the game to machines with more than two symbols [7], a new metric for classifying machines was used. Instead of Radós original BB-n classification, we now classify a BBG entrant as a machine BB(n, k) which is a Turing machine with n states and k symbols.

Present Landscape. At the time of writing, 4 classes of busy beaver machines have had confirmed S and Σ scores with machines to match: BB(1,2), BB(2,2) BB(3,2) and BB(4,2). Marxen and Buntrock [6] have established lower bounds for BB(5,2) at $S(5,2) \geq 47,176,870$ and $\Sigma(5,2) \geq 4098$.

The father and son team of Terry and Shawn Ligocki have made progress in exploring the space of machines with more than 2 symbols by using simulated annealing techniques to obtain high scoring machines. They currently hold the record for many of these classes.

Table 1, compiled by Pascal Michel [7] shows the current records for a few of the classes as of June 2012.

Table 1. Currently known lower bounds of the explored classes.

Date	**Discoverer(s)**	**Bounds**
1963	Radó, Lin	$S(2,2) = 6,\ \Sigma(2,2) = 4$ $S(3,2) = 21,\ \Sigma(3,2) = 6$
1964	Brady	$S(4,2) = 107,\ \Sigma(4,2) = 13$
February 1990	Marxen, Buntrock	$S(5,2) \geq 47,176,870,\ \Sigma(5,2) \geq 4098$
February 2005	T. and S. Ligocki	$S(2,4) \geq 40,737,\ \Sigma(2,4) \geq 3,932,964$
November 2007	T. and S. Ligocki	$S(3,3) \geq 119,112,334,170,342,540, \Sigma(3,3) \geq 374,676,383$ $S(2,5) > 1.9 \times 10^{704}$,$\Sigma(2,5) > 1.7 \times 10^{352}$
December 2007	T. and S. Ligocki	$S(3,4) > 5.2 \times 10^{13036}, \Sigma(3,4) > 3.7 \times 10^{6518}$
January 2008	T. and S. Ligocki	$S(4,3) > 1 \times 10^{14072}, \Sigma(4,3) > 1.3 \times 10^{7936}$ $S(2,6) > 2.4 \times 10^{9866}, \Sigma(2,6) > 1.9 \times 10^{4933}$
June 2010	Kropitz	$S(6,2) > 7.4 \times 10^{36534}, \Sigma(6,2) > 3.4 \times 10^{18267}$

3 The Random Access Stored Program Model

The Random Access Stored Program (RASP) machine model from Elgot and Robinson [4], is a computational model similar to the Random Access Memory

(RAM) machine but with the distinction that the internal variables of the state machine, as well as the program itself, is stored in memory along with the data.

This idea that the machine only has a single block of memory means that a program can potentially read and write to any address in the memory – which allows for programs which modify themselves while they are being executed. The

A RASP machine is arranged as an array M of size l, where each location (or register) in M can hold a single natural number and each register is referred to by a natural number address. The state machine uses the first 3 registers for specific tasks – although that doesn't prevent any external read/write operation to take place. The top 3 registers of M are defined to be: the program counter (PC), the instruction register (IR), and the accumulator (ACC). The PC contains the memory address of the current instruction, the IR is used for decoding the instruction, and the ACC is the register which the numerical instructions modify and which is tested by the conditional instruction.

We can define the RASP machine to be either bounded or unbounded depending on how concrete we desire the instance of the model to be. An unbounded RASP machine is of an arbitrary size and each register can hold a natural number of unbounded size. The bounded RASP is specified in n bits, the size of the machine is set as 2^n registers numbered 0 to $2^n - 1$, each register can also hold a natural number x in the range $0 \leq x < 2^n$.

An equivalent definition is as a pair (N, K), where N is the number of registers in the machine and K is the maximum integer that can be expressed. This allows us to easily define RASP machines with unbounded registers which can hold finite integers (∞, K), or finite registers which themselves are unbounded (N, ∞). For the rest of this paper – unless otherwise stated – RASP "n-bit machines" will be of the form $(2^n, 2^n - 1)$

In the bounded RASP, over and underflow of a register is not treated as a special case. For example, when the machine executes the final instruction in the memory, the next increment of the PC (currently set to $2^n - 1$) will overflow it to 0 and the running of the machine continues as normal.

RASP Instructions and Programs. A RASP machine program is a sequence of natural numbers of length $\leq 2^n - 3$. When the machine is instantiated, the top 3 registers are set to (3 0 0), which is a zeroed IR and ACC and the PC is pointing to the first line of the program.

The execution of instructions by the machine follows the standard fetch-decode-execute cycle:

1. Copy into the IR, the instruction contained in the memory location pointed to by the contents of the PC.
2. Read the IR and decode which instruction the contents refer to.
   (a) If the instruction is not recognised – halt.
   (b) If the instruction takes a parameter – increment the PC and repeat step 1 to obtain it.
3. Execute the instruction.
4. Increment the PC, if the PC becomes equal to K – overflow to 0.

The instructions that can be executed by the machine are listed below.

- HALT – Halt execution.
- INC – Increment the accumulator.
- DEC – Decrement the accumulator.
- LOAD c – Load the value c into the accumulator.
- STO m – Store the value of the accumulator in register m.
- JGZ m – Set the PC to the value m if the accumulator is greater than zero.
- OUT – print the character '1'.
- CPY m – Copy the contents of register m into the accumulator.

LOAD, STO, JGZ, and CPY also require an operand which is assumed to be located in the proceeding memory location. In the case of these instructions, another fetch is performed in order to retrieve the operand. If the RASP machine attempts to interpret an integer which isn't an instruction, it halts as if it has executed the HALT instruction.

The state machine for the RASP machine uses a map to match an opcode to one of the above instructions. This allows us to produce arbitrary injective mappings between instructions and the natural numbers. The instruction set mapping is defined as a map $M : \mathbb{N} \mapsto \mathbb{INS}$ where
$\mathbb{INS} = \{HALT, INC, DEC, LOAD, STO, OUTJGZ, CPY\}$.

4 RASP as a Busy Beaver

In a RASP machine, the command 'OUT' can be thought of as printing a symbol to an attached screen, or writing to a write-only output tape. This idea invites us to draw comparisons between the σ and s functions for RASP and TMs. With a RASP machine R, we define $\sigma(R)$ to be the number of times 'OUT' has been executed and $s(R)$ as the number of fetch-decode-execute cycles that have been performed in total. In addition to this, the number of bits we specify for the bounded RASP machine defines natural classes of machines for the competition.

We define an entry into the BBG RASP (BB-$R(n)$) competition, as a pair (P, IS) where P is the program of size n and IS is the instruction set mapping.

Observations on RASP Machines. In using this model to play the BBG, we can make some observances on the nature of the machines which provide some insight into how easy searching for the champion machines in a class will be.

Theorem 1. *The halting problem for the bounded RASP machine is decidable.*

Proof. Consider a bounded n-bit RASP machine M. We define the state of M to be the entire memory at a particular time, and each fetch-decode-execute cycle as a transition from one state to another. Since there is only a finite range of values for a finite number of memory locations, we can calculate the maximum number of possible states for any given machine $numStates(n) = N^N$.

Because each fetch-decode-execute cycle performs a transition between states $S \to S'$ we can run the machine for $numStates(n)$ cycles before concluding that for some state X which is entered during execution of the machine, there exists a transitive closure over a relation R such that XR^+X. From which we can conclude that M will never halt. □

In practice, we rarely need to run a machine for $numStates(n)$ steps before we can work out if it halts or not. It suffices to store each visited state as it is encountered and check the store for the new state after every state transition, if we encounter the same state twice, a loop has occurred. Different instruction set mappings behave differently; because RASP programs are simultaneously programs and data – the machine can attempt to execute any part of the memory as if it were all instructions, and it can read/write to any part of the memory as if it were all data.

This means that the magnitude of the value which represents certain instructions can affect the s or σ value of the machines. As an example, consider tables 2 and 3.

The program in table 2 is the best program that can be found for the specified instruction set. It has scores of $S(3) \geq 33$ and $\Sigma(3) \geq 16$. Table 3 is the champion machine for $BB - R(3)$ which was found through brute force searching of the entire space of program/instruction set combinations and it presents with scores of $S(3) = 82$ and $\Sigma(3) = 47$.

Table 2. The best 3-bit machine with the instruction set $\{0 \mapsto HALT, 1 \mapsto INC, 2 \mapsto DEC, 3 \mapsto LOAD, 4 \mapsto STO, 5 \mapsto OUT, 6 \mapsto JGZ, 7 \mapsto CPY\}$

Memory Address(es)	**Integer**	**Instruction**	**Label**
0	3		:PC
1	0		:IR
2	0		:ACC
3	1	INC	:start
4	5	OUT	
5	5	OUT	
6	6	JGZ	
7	3	LOAD	

The number of individual machines arising from these program/instruction set combinations grows very quickly. The number of unique instruction sets for a given n-bit machine is

$$\prod_{N-8<i\leq N} i \tag{1}$$

To calculate the number of programs, we recognise that the IR and ACC are initialised to 0 and that the PC points to the first command in the program in memory address 3. Since these are constant in the initial state of every program, we need to calculate the number of possible base N numbers of length $N - 3$.

If we consider each register to contain a digit of the base N number, the number of possible machines with a starting state of (3 0 0) comes to $(2^n)^{2^n-3}$.

Table 3. The best 3-bit machine with the instruction set $\{0 \mapsto OUT, 1 \mapsto LOAD, 2 \mapsto DEC, 3 \mapsto INC, 4 \mapsto CPY, 5 \mapsto STO, 6 \mapsto HALT, 7 \mapsto JGZ\}$

Memory Address(es)	**Integer**	**Instruction**	**Label**
0	3		:PC
1	0		:IR
2	0		:ACC
3	3	INC	:start
4	3	INC	
5	0	OUT	
6	0	OUT	
7	3	INC	

Like the number of possible instruction sets, this number grows quickly. Indeed, the space of machines becomes rapidly intractable for all but the smallest machines ($n < 4$). Combining these two values demonstrates the intractability of brute forcing a result. As an example, for 4 bit machines

$$\begin{aligned}(2^4)^{2^4-3} \times \prod_{8<i\leq 16} i &= 4,503,599,627,370,496 \times 518,918,400 \\ &= 2,337,000,712,875,693,991,526,400 \end{aligned} \tag{2}$$

5 Searching for the Best

A parallelised version of the brute force algorithm was implemented to find the champion machine for 3-bit machines which is shown in table 3. The brute force algorithm generates all the possible instruction sets and dispatches even chunks of these to the workers. Once a worker has determined their best machine, it is returned to the master which determines the very best and returns it. This works for finding the best 3-bit machine however is hopelessly inadequate in tackling the sheer volume of cases which arises from using more than 3 bits. A genetic algorithm (GA) [5] approach was therefore employed so that we can use an informed search to explore the space of machines.

There are a myriad of methods with which to explore the space, however we have observed that the search space tends to be very volatile. A neighbour solution which differs by a single instruction or a slight change to the instruction set mappings can produce very different results. As such (meta)heuristic methods of searching the space – for instance, local search – were though to be too sensitive to these changes. A more global search method was decided upon as genetic algorithms. Simulated annealing – as employed by the Ligockis – would also have been an acceptable alternative and experimentation with that method .

Overview of the Algorithm. The algorithm initialises a random pool of n-bit machines fitting the constraints for RASP machines set out above and proceeds

to determine their fitness. The fitness function is the number of OUT commands executed, but if the machine is shown not to halt (by using the loop detection strategy mentioned after the above decidability proof) then the fitness of the machine is defined to be 0.

Once the machines have all been executed, roulette selection [5] is performed to select a group of machines to populate the next generation. The machines not selected are removed from the pool, and the selected ones are preserved and randomly crossed in order to produce new machines to fill the pool.

Crossing involves first picking two parents, then a method of reproduction – crossing program only, instruction set only, or both. In the case of not crossing both, a 'dominant' parent is randomly selected which is used to fill in the non-crossed attribute of the machine by directly copying it into the machine.

The crossing itself is done by picking a point on the program or instruction set – which are our chromosomes represented as a vector in memory – and combining the left side of the chromosome from parent A (which is arbitrarily chosen) with the right side of the chromosome from parent B. In the case of the instruction set, if there is a duplicate entry such that the map is no longer injective then substitute integers are randomly generated for the duplicates until the map is injective again.

There is also a random chance of mutation to the instruction set or program which will pick two values in the program or instruction set of the child machine and swap their positions.

Parallelisation of the algorithm was through means of an 'isolated island' [5] approach where each processor in the computation has their own pool and returns their best found machine to the master processor which sorts them and returns the overall best of the computation.

For the 4 and 5 bit machines, the generation of high scoring machines was helped along by seeding the pool with a previously found high scoring machine at the pool initialisation stage. It became a good strategy to seed the current champion so that the algorithm can attempt to improve on it.

Current Results. The parallel brute force algorithm calculating the best 3-bit machine was benchmarked on a lightly loaded dual Intel Xeon E5506 machine running at 2.13GHz. This setup provides 8 cores which were apportioned as 7 worker processors and a master.

The times (seconds) taken across 3 runs were 1740, 1743, and 1678 for an average runtime of 1719 seconds. Each of these machines were executed for an average of 5 fetch-execute cycles each. This is due to the loop detection system. It appears that most RASP machines either halt or enter an infinite loop early – which is similar to the findings of [2] whereby most programs either halt quickly or loop forever. From a random sampling of 8,402,100,000 4 bit-machines, our average number of fetch execution cycles is 1.8 which translates into around 305,045 machines per second.

This is a very rough approximation as the sample size is tiny compared to the space. Given our calculated number of 4 bit machines in (2), we can see that

this would take a long time - assuming 256 workers, it will take approximately 948 million years - which underlines our need for informed search methods.

Table 4. Current records for numbers of shifts and outputs.

Bits	Instruction Set	Results	Comments
3	{6,3,2,1,5,0,7,4}	$S(3) = 82, \Sigma(3) = 47$	Exact values found through brute force searching.
4	{2,6,7,5,1,3,4,0}	$S(4) \geq 2668, \Sigma(4) \geq 1483$	Genetic, Pool: 100000, Generations: 1000, Islands: 32
5	{7,1,0,2,4,3,6,5}	$S(5) \geq 7540865, \Sigma(5) \geq 5242881$	Genetic, Pool: 100000, Generations: 500, Islands: 32
5	{4,7,1,3,5,2,6,0}	$S(5) \geq 235652, \Sigma(5) \geq 163846$	This machine was constructed by hand using 3 nested loops.

The experiment confirms that the number of machines grows very rapidly with the size of the machine word The experiment also confirms that the largest BBG solution found grows very rapidly with the size of machine word.

The best 5 bit RASP found by the GA (5,242,881) is 50 times better than a good seed machine with 3 nested loops constructed by hand (163,846). From inspection, the evolved machine makes considerable use of self-modification, with programming constructs which are very hard to characterise succinctly.

6 Conclusions, Reflections and Further Work

We have introduced our RASP model variant and ported the BBG over to it. From the current champions, we can see that they heavily leverage the ability of RASP machines to self-modify in order to produce high shift and output counts.

The halting problem for the bounded RASP machine has been shown to be decidable, but since both the size of the machine and the range of natural numbers expressible by the machine double with each successive class; the space of possible champions explodes. This renders exhaustive searching of all the candidates too time consuming to consider so we produced a genetic algorithm to perform an informed search for the champions.

The resulting GA has found and improved on candidates for the champion machines in the 4 and 5 bit classes. However the current GA tends to be weak with respect to the diversity of the gene pool. Thus, the current best machine tends to dominate the pool within a few generations which leads to the GA only attempting variations of this machine and mostly finding highly local optima.

Adapting the reproduction and mutation mechanics can allow for more diversity (i.e. reproduction eliminates breeding parents after replenishing the pool). The isolated island approach also unduly constrained the search procedure. Using a 'true' island approach, diversity can be introduced through migration of a few best scoring machines to a limited number of adjacent processes.

Aside from improvements to the GA, finding optimal BBG solutions for RASP machines can benefit from methods of classifying high scoring or non-halting machines. There may be distinguishing patterns of machines which can augment a brute force approach by allowing us to discount large numbers of machines as being suboptimal or non-halting. The observation that a large number of non-halting machines are shown to quickly enter loops implies there is a pattern which we can discern and use to our advantage.

We now have BBG expressed in TMs and RASP machines, and parallel GA machinery to explore both spaces. We have also implemented a Universal TM and a RASP interpreter as both TMs and RASP machines. We next plan to construct compilers from TM to RASP and from RASP to TM in both TM and RASP. This will enable us to explore the comparative elegance of TM and RASP programs, and Chaitin's LISP programs, taking into account all layers of language realisation down to the formal semantics in a common notation.

We would also like to use GA techniques to look for paradigmatic programming patterns in high scoring RASP machines.

Acknowledgements. We would like to thank our colleague Phil Trinder for his constructive reflections on this research, and to the 4 anonymous reviewers whose comments have improved the quality of this paper.

Joe Davidson is pleased to acknowledge the support of the School of Mathematical and Computer Science, Heriot-Watt University for his PhD study.

References

1. Allen H. Brady. The busy beaver game and the meaning of life. In Rolf Herken, editor, *The universal Turing machine (2nd ed.)*, pages 237–254. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1995.
2. Cristian S. Calude and Michael Stay. Most programs stop quickly or never halt. *CoRR*, abs/cs/0610153, 2006.
3. Gregory J. Chaitin. *The Limits of Mathematics : A Course on Information Theory and the Limits of Formal Reasoning (Discrete Mathematics and Theoretical Computer Science)*. Springer, October 2002.
4. Calvin C. Elgot and Abraham Robinson. Random-access stored-program machines, an approach to programming languages. *J. ACM*, 11(4):365–399, 1964.
5. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.
6. H. Marxen and J. Buntrock. Attacking Busy Beaver 5. *Bulletin of the European Association for Theoretical Computer Science*, 40, 1990.
7. Pascal Michel. The busy beaver competition: a historical survey, 2012.
8. T. Rado. On non-computable functions. *The Bell System Technical Journal*, 41(3):877–884, 1962.

An Extended Fundamental Duality of Partially Ordered Sets and Its Applications

Mustafa Demirci

Akdeniz University, Faculty of Sciences,
Department of Mathematics, 07058 Antalya, Turkey
demirci@akdeniz.edu.tr

Abstract. For a fixed augmented partially ordered set A, it is shown in this paper that there exists a dual equivalence between the category of A-spatial augmented partially ordered sets and the category of A-sober A-valued spaces. Then, as its application, for a fixed $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete partially ordered set L, we have established a dual equivalence between the category of L-spatial $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete partially ordered sets and the category of L-sober L-valued $\mathcal{Q}$-spaces. Furthermore, some concrete applications of the results to many familiar categories are given.

Key words: Poset, Subset system, Augmented poset, Poset-valued set, $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete poset, Poset-valued space

1 Introduction

The classification of partially ordered sets referring to certain specified joins and meets has been a major issue in various fields of order theory [6, 8, 10, 11, 19], algebra [3, 17], computer science [1, 16, 21] and topology [7–10, 12]. There are two main approaches to this classification. The first one involves the notion of subset selection [6, 8–10], that is, a rule $\mathcal{Z}$ assigning to each partially ordered set (poset for short) P a subset $\mathcal{Z}(P)$ of the power set $\mathcal{P}(P)$ of P, and is unified under the framework of $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete posets in [5]. Here the subset selection generalizes the subset system, originally introduced by Wright et. al. [21]. For a quadruple $\mathcal{Q} = (\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_3, \mathcal{Z}_4)$ of subset selections $\mathcal{Z}_1$, $\mathcal{Z}_2$ and subset systems $\mathcal{Z}_3$, $\mathcal{Z}_4$, $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete posets form a category $\mathcal{Q}\mathbf{P}$ of $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete posets. The second one, which is less popular than the first one, is proposed by Banaschewski and Bruns [4]. Augmented posets play a key role in their approach, and constitute a category $\mathbf{P}$. The adjunction $T \dashv \Psi : \mathbf{P}^{op} \to \mathbf{S}$ between the opposite $\mathbf{P}^{op}$ of $\mathbf{P}$ and the category $\mathbf{S}$ of spaces is one of the significant contributions in [4], while the dual equivalence between the full subcategory $\mathbf{SpaP}$ of $\mathbf{P}$ of all spatial objects and the full subcategory $\mathbf{SobS}$ of $\mathbf{S}$ of all sober objects is another one. As a topological counterpart of $\mathcal{Q}\mathbf{P}$, $\mathcal{Q}$-spaces and their category $\mathcal{Q}\mathbf{S}$ have been introduced in [5]. Furthermore, by establishing two full embeddings $G_{\mathcal{Q}} : \mathcal{Q}\mathbf{P} \to \mathbf{P}$ and $H_{\mathcal{Q}} : \mathcal{Q}\mathbf{S} \to \mathbf{S}$, it is demonstrated in [5] that the latter is category-theoretically more general approach than the former, while the former

yields feasible results in applications. Despite the fact that both approaches provide useful and powerful tools to unify various kinds of generalized topological spaces under the same framework, they are inadequate to handle the problems related with the poset-valued analogous of generalized topological spaces which result from the essence of fuzzy logic [15]. Representations of posets by poset-valued base spaces, of complete lattices by lattice-valued closure spaces and of directed-complete posets by lattice-valued algebraic closure spaces are examples of such problems. To overcome this shortcoming, we introduce poset-valued spaces and poset-valued $\mathcal{Q}$-spaces, and extend all central results of [4, 5] to the present approach. More specifically, referring to a fixed augmented poset A and a fixed $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete poset L, we extend $\mathbf{S}$ and $\mathcal{Q}\mathbf{S}$ to the category A-$\mathbf{S}$ of A-valued spaces and the category L-$\mathcal{Q}\mathbf{S}$ of L-valued $\mathcal{Q}$-spaces, respectively. As a generalization of $T \dashv \Psi : \mathbf{P}^{op} \to \mathbf{S}$, the adjunction $AT \dashv A\Psi : \mathbf{P}^{op} \to A$-$\mathbf{S}$ is proven in Theorem 1. The dual equivalence between the full subcategory of $\mathbf{P}$ of all A-spatial augmented posets and the full subcategory of A-$\mathbf{S}$ of all A-sober A-spaces, given in Corollary 1, extends the dual equivalence between $\mathbf{SpaP}$ and $\mathbf{SobS}$ to the A-valued spaces. The main result of [5] ([5, Theorem 2]) is re-formulated for L-valued $\mathcal{Q}$-spaces in Theorem 3 consisting of a dual adjunction between the full subcategory L-$\mathcal{Q}\mathbf{P}_s$ of $\mathcal{Q}\mathbf{P}$ of all L-$\mathcal{Q}$-spatial objects and L-$\mathcal{Q}\mathbf{S}$, $(\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_1, \mathcal{Z}_2)\mathbf{P}$ and L-$(\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_1, \mathcal{Z}_2)\mathbf{S}$ and a dual equivalence between L-$\mathcal{Q}\mathbf{P}_s$ and the full subcategory of L-$\mathcal{Q}\mathbf{S}$ of all L-$\mathcal{Q}$-sober objects under some reasonable assumptions on $\mathcal{Z}_1$, $\mathcal{Z}_2$ and $\mathcal{Q}$. In order to show the usefulness of the presented results, we give their direct applications to some familiar order-theoretic categories in Corollary 3.

2 Classification of Posets

2.1 Augmented Posets

An augmented poset is a triple $A = (|A|, \mathfrak{J}A, \mathfrak{M}A)$, consisting of a poset $|A|$, a subset $\mathfrak{J}A$ of $\mathcal{P}(|A|)$ in which each member has the join in $|A|$ and a subset $\mathfrak{M}A$ of $\mathcal{P}(|A|)$ in which each member has the meet in $|A|$. Augmented posets together with structure preserving maps constitute a category $\mathbf{P}$ [4]. A structure preserving map $h : A \to B$ between the augmented posets A and B here means a monotone map $h : |A| \to |B|$ such that $h(S) \in \mathfrak{J}B$ and $h(\bigvee S) = \bigvee h(S)$ for all $S \in \mathfrak{J}A$, and $h(R) \in \mathfrak{M}B$ and $h(\bigwedge R) = \bigwedge h(R)$ for all $R \in \mathfrak{M}A$ [4]. For each set X and an augmented poset A, the X-th power of A is the augmented poset A^X such that $|A^X| = |A|^X$, $\mathfrak{J}A^X$ is the set of subsets $S \subseteq |A|^X$ with the property that the image of S under the x-th projection map $\pi_x : |A|^X \to |A|$ belongs to $\mathfrak{J}A$ for all $x \in X$, and analogously for $\mathfrak{M}A^X$ [4]. Due to the terminology of Goguen [14], the elements of $|A|^X$ are called $|A|$-sets, generalizing fuzzy sets [22].

2.2 $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete Posets

A subset selection $\mathcal{Z}$ [6, 8–10] is, by definition, a class-theoretic function sending each poset P to a set $\mathcal{Z}(P)$ of subsets of P whose elements are the so-called

$\mathcal{Z}$-sets of P. A subset selection $\mathcal{Z}$ is called a subset system [17, 21] if for each monotone map $f : P \to Q$, $M \in \mathcal{Z}(P)$ implies $f(M) \in \mathcal{Z}(Q)$. It is said that a subset selection $\mathcal{Z}$ preserves surjectivity if for each surjective monotone map $f : P \to Q$ and for each $M \in \mathcal{Z}(Q)$, there exists at least one $N \in \mathcal{Z}(P)$ such that $M = f(N)$. Throughout this paper, $\mathcal{Z}$ and $\mathcal{Z}_i$ $(i = 1, ..., 4)$ always stand for subset selections if further assumptions are not made explicitly. In this paper, as examples for the subset selections that are subset systems at the same time, we only consider $\mathcal{Z} = \mathcal{V}$, $\mathcal{F}$, $\mathcal{D}$, $\mathcal{C}n$, $\mathcal{P}$, where $\mathcal{Z}$-sets of each poset P are no subset of P, finite subsets of P, directed subsets of P, countable subsets of P and all subsets of P, respectively. Note that all of them except $\mathcal{D}$ are surjectivity-preserving.

A poset P is called $\mathcal{Z}$-$\bigvee(\bigwedge)$-complete iff each $M \in \mathcal{Z}(P)$ has a join (meet) in P [3, 8, 9, 17, 21]. A $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete poset is defined to be a $\mathcal{Z}_1$-$\bigvee$-complete and $\mathcal{Z}_2$-$\bigwedge$-complete poset [5]. If we associate two subset selections $\mathcal{Z}^{\sup}$ and $\mathcal{Z}^{\inf}$ to $\mathcal{Z}$ such that $M \in \mathcal{Z}^{\sup(\inf)}(P)$ iff $M \in \mathcal{Z}(P)$ with $\bigvee M$ $(\bigwedge M)$ in P, then P is $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete iff $\mathcal{Z}_1(P) = \mathcal{Z}_1^{\sup}(P)$ and $\mathcal{Z}_2(P) = \mathcal{Z}_2^{\inf}(P)$.

A monotone function $f : P \to Q$ is $\mathcal{Z}$-$\bigvee$-continuous iff for each $M \in \mathcal{Z}^{\sup}(P)$, $\bigvee f(M) = f(\bigvee M)$, and is $\mathcal{Z}$-$\bigwedge$-continuous iff for each $M \in \mathcal{Z}^{\inf}(P)$, $\bigwedge f(M) = f(\bigwedge M)$. A function, which is $\mathcal{Z}_1$-$\bigvee$-continuous and $\mathcal{Z}_2$-$\bigwedge$-continuous simultaneously, is called $(\mathcal{Z}_1, \mathcal{Z}_2)$-continuous. Whenever $\mathcal{Z}_3$ and $\mathcal{Z}_4$ are taken as subset systems, $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete posets and $(\mathcal{Z}_3, \mathcal{Z}_4)$-continuous maps constitute a category $\mathcal{Q}\mathbf{P}$, where $\mathcal{Q} = (\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_3, \mathcal{Z}_4)$ [5]. For the sake of shortness, any occurrence of $\mathcal{Q} = (\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_1, \mathcal{Z}_2)$ will be replaced by the pair $(\mathcal{Z}_1, \mathcal{Z}_2)$ in this paper, e.g. $(\mathcal{Z}_1, \mathcal{Z}_2, \mathcal{Z}_1, \mathcal{Z}_2)\mathbf{P}$ will be shortly written as $(\mathcal{Z}_1, \mathcal{Z}_2)\mathbf{P}$. It is shown in [5] that most of the familiar order-theoretic constructs can be expressed in the form of $\mathcal{Q}\mathbf{P}$. Now we give a few examples of $\mathcal{Q}\mathbf{P}$ that will be used subsequently, and refer reader to [5] for many other examples and details.

Example 1. For $\mathcal{Q} = (\mathcal{V}, \mathcal{V}, \mathcal{V}, \mathcal{V})$ ($\mathcal{Q} = (\mathcal{F}, \mathcal{F}, \mathcal{F}, \mathcal{F})$, $\mathcal{Q} = (\mathcal{P}, \mathcal{F}, \mathcal{P}, \mathcal{F})$, $\mathcal{Q} =(\mathcal{V}, \mathcal{P}, \mathcal{V}, \mathcal{P})$, $\mathcal{Q} = (\mathcal{F}, \mathcal{P}, \mathcal{F}, \mathcal{P})$, $\mathcal{Q} = (\mathcal{D}, \mathcal{P}, \mathcal{D}, \mathcal{P})$, $\mathcal{Q} = (\mathcal{P}, \mathcal{P}, \mathcal{P}, \mathcal{P})$, $\mathcal{Q} =(\mathcal{C}n, \mathcal{F}, \mathcal{C}n, \mathcal{F})$, $\mathcal{Q} = (\mathcal{D}, \mathcal{F}, \mathcal{D}, \mathcal{F})$), $\mathcal{Q}\mathbf{P}$ is known as the category **Pos** of posets and monotone maps [2] (the category **Blatt** of bounded lattices and maps preserving finite joins and and finite meets [4], the category $\mathbf{SUP}^{\wedge}$ of complete lattices and maps preserving arbitrary joins and finite meets [13], the category **MCPos** of complete lattices and maps preserving arbitrary meets [2], the category $\mathbf{INF}^{\vee}$ of complete lattices and maps preserving arbitrary meets and finite joins [5], the category $\mathbf{INF}^{\uparrow}$ of complete lattices and maps preserving arbitrary meets and directed joins [13], the category **CLat** of complete lattices and maps preserving arbitrary meets and arbitrary joins [2], the category σ**ComLat** of σ-complete lattices and maps preserving countable joins and finite meets [5], the category **QF** of quasiframes and Scott-continuous functions preserving finite meets [11], resp.).

Since the functor $G_{\mathcal{Q}} : \mathcal{Q}\mathbf{P} \to \mathbf{P}$, defined by $G_{\mathcal{Q}}(P) = \left(P, \mathcal{Z}_3^{\sup}(P), \mathcal{Z}_4^{\inf}(P)\right)$ and $G_{\mathcal{Q}}(f) = f$, is a full embedding [5], $\mathbf{P}$ can be interpreted as a category larger than $\mathcal{Q}\mathbf{P}$. Thus, the notion of augmented poset forms a more general approach

than the notion of $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete poset for the classification of posets via their existing joins and existing meets.

3 Duality Between Augmented Posets and Poset-Valued Spaces

3.1 Category of Poset-Valued Spaces

In the formulation of the categories of spaces [4] and of poset-valued spaces, for a given function $f : X \to Y$ between the sets X and Y, and for a subset U (V) of X (Y), we denote by $f^{\rightarrow}(U)$ $(f^{\leftarrow}(V))$ the image (preimage)) of U (V) under f.

Proposition 1. *Let A be an augmented poset. Each function $f : X \to Y$ induces a $\boldsymbol{P}$-morphism $f_A^{\leftarrow} : A^Y \to A^X$, defined by $f_A^{\leftarrow}(\mu) = \mu \circ f$ for each $\mu \in |A|^Y$.*

A space is a quadruple $W = (|W|, \mathfrak{O}(W), \Sigma(W), \Delta(W))$, where $|W|$ is a set, $\mathfrak{O}(W)$ is a subset of $\mathcal{P}(|W|)$, $\Sigma(W)$ is a subset of $\{\mathfrak{U} \subseteq \mathfrak{O}(W) \mid \bigcup \mathfrak{U} \in \mathfrak{O}(W)\}$ and $\Delta(W)$ is a subset of $\{\mathfrak{V} \subseteq \mathfrak{O}(W) \mid \bigcap \mathfrak{V} \in \mathfrak{O}(W)\}$. Spaces form a category $\mathbf{S}$ [4] whose morphisms $f : W_1 \to W_2$ are functions $f : |W_1| \to |W_2|$ satisfying the conditions that $(f^{\leftarrow})^{\rightarrow}(\mathfrak{O}(W_2)) \subseteq \mathfrak{O}(W_1)$, $(f^{\leftarrow})^{\rightarrow}(\mathfrak{U}) \in \Sigma(W_1)$ for each $\mathfrak{U} \in \Sigma(W_2)$, and $(f^{\leftarrow})^{\rightarrow}(\mathfrak{V}) \in \Delta(W_1)$ for each $\mathfrak{V} \in \Delta(W_2)$.

In the following considerations, we fix an augmented poset A, and extend the category of spaces to the category of A-valued spaces.

Definition 1. *The category A-$\boldsymbol{S}$ consists of the following information: Objects are A-valued spaces (A-spaces for short), i.e. quadruples $Z = (|Z|, \mathfrak{O}_A(Z), \Sigma_A(Z), \Delta_A(Z))$, where $|Z|$ is a set, $\mathfrak{O}_A(Z)$ is a subset of $|A|^{|Z|}$, $\Sigma_A(Z)$ is a set of subsets $\mathfrak{U} \subseteq \mathfrak{O}_A(Z)$ such that $\mathfrak{U} \in \mathfrak{J}A^{|Z|}$ and $\bigvee \mathfrak{U} \in \mathfrak{O}_A(Z)$, and $\Delta_A(Z)$ is a set of subsets $\mathfrak{V} \subseteq \mathfrak{O}_A(Z)$ such that $\mathfrak{V} \in \mathfrak{M}A^{|Z|}$ and $\bigwedge \mathfrak{V} \in \mathfrak{O}_A(Z)$, while morphisms $f : Z_1 \to Z_2$ are functions $f : |Z_1| \to |Z_2|$ satisfying the properties that $(f_A^{\leftarrow})^{\rightarrow}(\mathfrak{O}_A(Z_2)) \subseteq \mathfrak{O}_A(Z_1)$, $(f_A^{\leftarrow})^{\rightarrow}(\mathfrak{U}) \in \Sigma_A(Z_1)$ for each $\mathfrak{U} \in \Sigma_A(Z_2)$, and $(f_A^{\leftarrow})^{\rightarrow}(\mathfrak{V}) \in \Delta_A(Z_1)$ for each $\mathfrak{V} \in \Delta_A(Z_2)$.*

Remark 1. The category A-$\mathbf{S}$ can also be equivalently defined by means of the category $\mathbf{P}$ of augmented posets as follows.

(i) A quadruple $Z = (|Z|, \mathfrak{O}_A(Z), \Sigma_A(Z), \Delta_A(Z))$, consisting of a set $|Z|$, a subset $\mathfrak{O}_A(Z)$ of $|A|^{|Z|}$, subsets $\Sigma_A(Z)$ and $\Delta_A(Z)$ of $\mathcal{P}(|A|^{|Z|})$, is an A-space iff $AT(Z) = (\mathfrak{O}_A(Z), \Sigma_A(Z), \Delta_A(Z))$ is an augmented poset and the inclusion map $i_{AT(Z)} : AT(Z) \hookrightarrow A^{|Z|}$ is a $\mathbf{P}$-morphism.

(ii) For two A-spaces Z_1, Z_2 and a map $f : |Z_1| \to |Z_2|$, $f : Z_1 \to Z_2$ is an A-$\mathbf{S}$-morphism iff the restriction of the $\mathbf{P}$-morphism $f_A^{\leftarrow} : A^{|Z_2|} \to A^{|Z_1|}$ to $AT(Z_2)$ yields a $\mathbf{P}$-morphism $(f_A^{\leftarrow})_{|_{AT(Z_2)}} : AT(Z_2) \to AT(Z_1)$.

The following proposition shows that $\mathbf{S}$ is a special case of A-$\mathbf{S}$.

Proposition 2. *For $2_{\mathbf{P}} = (2, \mathcal{P}(2), \mathcal{P}(2))$, $2_{\mathbf{P}}$-$\boldsymbol{S}$ is isomorphic to $\boldsymbol{S}$.*

Proof. Given a set X, $\mathfrak{U} \subseteq \mathcal{P}(X)$ and $\Phi \subseteq \mathcal{P}(\mathcal{P}(X))$, let $\mathfrak{U}^c$ denote the set of characteristic functions $\chi_V : X \to 2$ of all $V \in \mathfrak{U}$, and $\Phi^* = \{\mathfrak{U}^c \mid \mathfrak{U} \in \Phi\}$. Then, the functor $F : \mathbf{S} \to 2_{\mathbf{P}}\text{-}\mathbf{S}$, defined by $F(W) = (|W|, \mathfrak{O}(W)^c, \Sigma(W)^*, \Delta(W)^*)$ for each $W \in Ob(\mathbf{S})$, and $F(f) = f$ for each $f \in Mor(\mathbf{S})$, is an isomorphism, proving the claim.

3.2 Relations Between *A*-S and $\mathbf{P}^{op}$

In this subsection, our aim is to extend the adjunction $T \dashv \Psi : \mathbf{P}^{op} \to \mathbf{S}$ to an adjunction $AT \dashv A\Psi : \mathbf{P}^{op} \to A\text{-}\mathbf{S}$, and then to refine the latter adjunction to an equivalence, involving the notions of A-spatiality and A-sobriety.

Proposition 3. *The map $AT : A\text{-}\boldsymbol{S} \to \boldsymbol{P}^{op}$, sending each A-$\boldsymbol{S}$-morphism $f : Z_1 \to Z_1$ to $(f_A^{\leftarrow})^{op}_{|_{AT(Z_2)}} : AT(Z_1) \to AT(Z_2)$, is a functor.*

Proof. The assertion can be easily seen from Remark 1.

Lemma 1. *Let B be an augmented poset. For each $a \in |B|$, let us define the map $\Psi_a : \boldsymbol{P}(B, A) \to |A|$, $h \mapsto h(a)$. Then, $A\Psi(B)$, where $|A\Psi(B)| = \boldsymbol{P}(B, A)$, $\mathfrak{O}_A(A\Psi(B)) = \{\Psi_a \mid a \in |B|\}$, $\Sigma_A(A\Psi(B))$ and $\Delta_A(A\Psi(B))$ are the sets of all sets $\{\Psi_a \mid a \in S\}$ such that $S \in \mathfrak{J}B$, respectively $S \in \mathfrak{M}B$, is an A-space.*

Proof. It is clear that $\mathfrak{O}_A(A\Psi(B)) \subseteq |A|^{\mathbf{P}(B,A)}$. For each $\mathfrak{U} \in \Sigma_A(A\Psi(B))$, there exists $S \in \mathfrak{J}B$ such that $\mathfrak{U} = \{\Psi_a \mid a \in S\}$. By considering the definitions of $\mathfrak{J}A^{\mathbf{P}(B,A)}$ and Ψ_a, one can easily observe that $\mathfrak{U} \in \mathfrak{J}A^{\mathbf{P}(B,A)}$ and $\bigvee \mathfrak{U} = \bigvee_{a \in S} \Psi_a = \Psi_{\bigvee S} \in \mathfrak{O}_A(A\Psi(B))$; therefore $\Sigma_A(A\Psi(B))$ is a subset of $\left\{\mathfrak{U} \subseteq \mathfrak{O}_A(A\Psi(B)) \mid \mathfrak{U} \in \mathfrak{J}A^{|A\Psi(B)|} \text{ and } \bigvee \mathfrak{U} \in \mathfrak{O}_A(A\Psi(B))\right\}$, and analogously for $\Delta_A(A\Psi(B))$. Thus, $A\Psi(B))$ is an A-space.

Proposition 4. *The map $A\Psi : \boldsymbol{P}^{op} \to A\text{-}\boldsymbol{S}$, defined by*

$$A\Psi\left(B_1 \overset{u}{\to} B_2\right) = A\Psi(B_1) \overset{A\Psi(u)}{\to} A\Psi(B_2),$$

where $[A\Psi(u)](h) = h \circ u^{op}$ for all $h \in |A\Psi(B_1)|$, is a functor.

Proof. Lemma 1 shows that $A\Psi$ defines a function from $Ob(\mathbf{P}^{op})$ to $Ob(A\text{-}\mathbf{S})$. Given each $\mathbf{P}^{op}$-morphism $B_1 \overset{u}{\to} B_2$, one can easily prove that $A\Psi(u) : A\Psi(B_1) \to A\Psi(B_2)$ is an A-$\mathbf{S}$-morphism. $A\Psi$ obviously preserves composition and the identities. Hence, $A\Psi$ is a functor.

Theorem 1. *$AT \dashv A\Psi : \boldsymbol{P}^{op} \to A\text{-}\boldsymbol{S}$.*

Proof. For each A-space W, define the map $\eta_W : |W| \to \mathbf{P}(AT(W), A)$ by $[\eta_W(x)](h) = h(x)$ for all $x \in |W|$, $h \in |AT(W)|$. Then, it is easy to see that $\eta_W : W \to A\Psi(AT(W))$ is an A-$\mathbf{S}$-morphism. Moreover, η_W is the W-th component of a natural transformation $\eta : id_{A\text{-}\mathbf{S}} \to A\Psi AT$. On the other hand, we may

also consider another natural transformation $\varepsilon : AT A\Psi \to id_{\mathbf{P}^{op}}$ whose each B-th component is the opposite of the $\mathbf{P}$-morphism $\varepsilon_B^{op} : B \to AT(A\Psi(B))$, defined by $\varepsilon_B^{op}(a) = \Psi_a$ for each $a \in |B|$. It is not difficult to verify that the following adjunction identities hold for each $B \in Ob(\mathbf{P})$ and $W \in Ob(A\text{-}\mathbf{S})$:

$$A\Psi(\varepsilon_B) \circ \eta_{A\Psi(B)} = id_{A\Psi(B)} \text{ and } \varepsilon_{AT(W)} \circ AT(\eta_W) = id_{AT(W)}.$$

Thus, $(\eta, \varepsilon) : AT \dashv A\Psi : \mathbf{P}^{op} \to A\text{-}\mathbf{S}$ is an adjoint situation, and so the assertion follows.

Definition 2. *(i) An augmented poset B is A-spatial iff the $\boldsymbol{P}$-morphism $\varepsilon_B^{op} : B \to AT(A\Psi(B))$, defined in Theorem 1, is an isomorphism in $\boldsymbol{P}$.*

(ii) An A-space W is A-sober iff the A-$\boldsymbol{S}$-morphism $\eta_W : W \to A\Psi(AT(W))$, defined in Theorem 1, is an isomorphism in A-$\boldsymbol{S}$.

Corollary 1. *Let A-$\boldsymbol{SpaP}$ be the full subcategory of $\boldsymbol{P}$ of all A-spatial augmented posets, and A-$\boldsymbol{SobS}$ be the full subcategory of A-$\boldsymbol{S}$ of all A-sober A-spaces. Then, the restrictions of $A\Psi$ and AT to A-$\boldsymbol{SpaP}^{op}$ and A-$\boldsymbol{SobS}$ induce equivalences $A\Psi_s : A\text{-}\boldsymbol{SpaP}^{op} \to A\text{-}\boldsymbol{SobS}$ and $AT_s : A\text{-}\boldsymbol{SobS} \to A\text{-}\boldsymbol{SpaP}^{op}$.*

Proof. The equivalences in question follow from Theorem 1 and [4, Lemma 1].

Without going into detail, it is worthwhile to mention here that $2_{\mathbf{P}}$-**SpaP** is the same as the full subcategory **SpaP** of **P** of all spatial augmented posets in [4], whereas $2_{\mathbf{P}}$-**SobS** coincides with the full subcategory **SobS** of **S** of all sober spaces in [4]. Thus, the dual equivalences between **SpaP** and **SobS** proven in [4, Proposition 2] are instances of Corollary 1. We conclude this section with the final remark that the co-domain of AT lies in A-$\mathbf{SpaP}^{op}$, while the co-domain of $A\Psi$ lies in A-**SobS**.

4 Poset-Valued $\mathcal{Q}$-spaces

Definition 3. *[5] The category $\mathcal{Q}\boldsymbol{S}$ consists of the following data: Objects are $\mathcal{Q}$-spaces (X, τ), that is, X is a set and τ is a subset of $\mathcal{P}(X)$ with the property that τ is a $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete poset ordered by set inclusion, and the inclusion map $i_\tau : \tau \hookrightarrow \mathcal{P}(X)$ is $(\mathcal{Z}_3, \mathcal{Z}_4)$-continuous. Morphisms $f : (X, \tau) \to (Y, \nu)$ are functions $f : X \to Y$ such that $(f^{\leftarrow})^{\rightarrow}(\nu) \subseteq \tau$.*

As a natural poset-valued extension of $\mathcal{Q}\mathbf{S}$, we introduce, for an arbitrarily fixed $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete poset L, the category L-$\mathcal{Q}\mathbf{S}$ of L-valued $\mathcal{Q}$-spaces, and point out in this section that L-$\mathcal{Q}\mathbf{S}$ can be fully embedded into $G_{\mathcal{Q}}(L)$-**S**. For this purpose, all components of $\mathcal{Q}$ are from now on assumed to be subset systems. Another important fact is that each function $f : X \to Y$ gives rise to a $\mathcal{Q}\mathbf{P}$-morphism $f_L^{\leftarrow} : L^Y \to L^X$, defined by $f_L^{\leftarrow}(\mu) = \mu \circ f$ for each $\mu \in L^Y$.

Definition 4. *The category L-$\mathcal{Q}\boldsymbol{S}$ comprises the following items: Objects are L-valued $\mathcal{Q}$-spaces, or shortly L-$\mathcal{Q}$-spaces (X, τ), i.e. X is a set and τ is a subset of L^X such that τ is a $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete poset equipped with the order inherited from L^X, and the inclusion map $i_\tau : \tau \hookrightarrow L^X$ is $(\mathcal{Z}_3, \mathcal{Z}_4)$-continuous. Morphisms $f : (X, \tau) \to (Y, \nu)$ are functions $f : X \to Y$ such that $(f_L^{\leftarrow})^{\rightarrow}(\nu) \subseteq \tau$.*

In a similar way to Proposition 2, one can easily observe that 2-$\mathcal{Q}$**S** is isomorphic to $\mathcal{Q}$**S**. The category L-$\mathcal{Q}$**S** enables us to formulate the categories of poset-valued extensions of various notions of spaces:

Example 2. (1) For a poset L and $\mathcal{Q} = (\mathcal{V}, \mathcal{V}, \mathcal{V}, \mathcal{V})$, L-$\mathcal{Q}$**S**, denoted by L-**BS**, is an extension of the category **BS** of base spaces, extensively studied in [10].

(2) For a bounded lattice L (i.e. L is a lattice with the top and bottom elements) and $\mathcal{Q} = (\mathcal{F}, \mathcal{F}, \mathcal{F}, \mathcal{F})$, L-$\mathcal{Q}$**S**, denoted by L-**BlatBS**, is an extension of the full subcategory **BlatBS** of **BS** in [5].

(3) For a complete lattice L and $\mathcal{Q} = (\mathcal{P}, \mathcal{F}, \mathcal{P}, \mathcal{F})$, L-$\mathcal{Q}$**S**, denoted by L-**Top**, is an extension of the category **Top** of topological spaces [2], and is also known as the category of L-topological spaces [20].

(4) For a complete lattice L and $\mathcal{Q} = (\mathcal{V}, \mathcal{P}, \mathcal{V}, \mathcal{P})$, L-$\mathcal{Q}$**S**, denoted by L-**CSp**, is an extension of the category **CSp** of closure spaces [12].

(5) For a complete lattice L and $\mathcal{Q} = (\mathcal{F}, \mathcal{P}, \mathcal{F}, \mathcal{P})$, L-$\mathcal{Q}$**S**, denoted by L-**TCSp**, is an extension of the category **TCSp** of topological closure spaces [12].

(6) For a complete lattice L and $\mathcal{Q} = (\mathcal{D}, \mathcal{P}, \mathcal{D}, \mathcal{P})$, L-$\mathcal{Q}$**S**, denoted by L-**ACSp**, is an extension of the category **ACSp** of algebraic closure spaces [12].

(7) For a complete lattice L and $\mathcal{Q} = (\mathcal{P}, \mathcal{P}, \mathcal{P}, \mathcal{P})$, L-$\mathcal{Q}$**S**, denoted by L-**ATSp**, is an extension of the category **ATSp** of Alexandroff-discrete spaces [12].

(8) For a σ-complete lattice L (i.e. L is a poset with countable joins and finite meets) and $\mathcal{Q} = (\mathcal{C}n, \mathcal{F}, \mathcal{C}n, \mathcal{F})$, L-$\mathcal{Q}$**S**, denoted by L-**Alex**, is an extension of the category **Alex** of Alexandroff spaces [4, 5].

(9) For a quasiframe L [11] (i.e. L is a poset with directed joins and finite meets) and $\mathcal{Q} = (\mathcal{D}, \mathcal{F}, \mathcal{D}, \mathcal{F})$, L-$\mathcal{Q}$**S**, denoted by L-**PreTop**, is an extension of the category **PreTop** of pretopological spaces [11].

The functor $H_{\mathcal{Q}} : \mathcal{Q}\mathbf{S} \to \mathbf{S}$, defined by

$$H_{\mathcal{Q}}(X, \tau) = \left(X, \tau, \mathcal{Z}_3^{\sup}(\tau), \mathcal{Z}_4^{\inf}(\tau)\right) \text{ and } H_{\mathcal{Q}}(f) = f,$$

is a full embedding [5]. We now extend this result to L-$\mathcal{Q}$-spaces.

Lemma 2. *Let X be a set, and let τ be a subset of L^X such that τ is a $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete poset equipped with the order inherited from L^X. Then, (X, τ) is an L-$\mathcal{Q}$-space iff $LH_{\mathcal{Q}}(X, \tau) = \left(X, \tau, \mathcal{Z}_3^{\sup}(\tau), \mathcal{Z}_4^{\inf}(\tau)\right)$ is a $G_{\mathcal{Q}}(L)$-space.*

Proof. Let (X, τ) be an L-$\mathcal{Q}$-space. $G_{\mathcal{Q}}(L)T(LH_{\mathcal{Q}}(X, \tau))$ is obviously an augmented poset. On the other hand, since $i_\tau : \tau \hookrightarrow L^X$ is a $\mathcal{Q}$**P**-morphism, $G_{\mathcal{Q}}(i_\tau) : G_{\mathcal{Q}}(\tau) \to G_{\mathcal{Q}}\left(L^X\right)$ is a **P**-morphism. Furthermore, we easily see that the identity map id_{L^X} on L^X is a **P**-morphism $G_{\mathcal{Q}}\left(L^X\right) \to G_{\mathcal{Q}}(L)^X$. Thus $i_\tau = id_{L^X} \circ G_{\mathcal{Q}}(i_\tau) : G_{\mathcal{Q}}(\tau) \to G_{\mathcal{Q}}(L)^X$ is a **P**-morphism. Then it follows from Remark 1 (i) that $LH_{\mathcal{Q}}(X, \tau)$ is a $G_{\mathcal{Q}}(L)$-space. Conversely, suppose $LH_{\mathcal{Q}}(X, \tau)$ is a $G_{\mathcal{Q}}(L)$-space. Then, $i_\tau : G_{\mathcal{Q}}(\tau) \to G_{\mathcal{Q}}(L)^X$ is a **P**-morphism by Remark 1 (i). To see that (X, τ) is an L-$\mathcal{Q}$-space, it is enough to confirm that the inclusion map $i_\tau : \tau \hookrightarrow L^X$ is $(\mathcal{Z}_3, \mathcal{Z}_4)$-continuous. For each $M \in \mathcal{Z}_3^{\sup}(\tau)$, since

$i_\tau : G_{\mathcal{Q}}(\tau) \to G_{\mathcal{Q}}(L)^X$ is a $\mathbf{P}$-morphism, we have $i_\tau(\bigvee M) = \bigvee i_\tau(M)$, i.e. i_τ is $\mathcal{Z}_3$-$\bigvee$-continuous. $\mathcal{Z}_4$-$\bigwedge$-continuity of i_τ is similar.

Theorem 2. *The functor $LH_{\mathcal{Q}} : L\text{-}\mathcal{Q}\boldsymbol{S} \to G_{\mathcal{Q}}(L)\text{-}\boldsymbol{S}$, defined by*

$$LH_{\mathcal{Q}}\left((X,\tau) \xrightarrow{f} (Y,\nu)\right) = LH_{\mathcal{Q}}(X,\tau) \xrightarrow{f} LH_{\mathcal{Q}}(Y,\nu),$$

is a full embedding.

Proof. We have from Lemma 2 that $LH_{\mathcal{Q}}$ sends each L-$\mathcal{Q}\mathbf{S}$-object to a $G_{\mathcal{Q}}(L)$-$\mathbf{S}$-object. Furthermore, each L-$\mathcal{Q}\mathbf{S}$-morphism $(X,\tau) \xrightarrow{f} (Y,\nu)$ yields a $G_{\mathcal{Q}}(L)$-$\mathbf{S}$-morphism $LH_{\mathcal{Q}}(X,\tau) \xrightarrow{f} LH_{\mathcal{Q}}(Y,\nu)$. Because $LH_{\mathcal{Q}}$ clearly preserves composition and the identities, $LH_{\mathcal{Q}} : L\text{-}\mathcal{Q}\mathbf{S} \to G_{\mathcal{Q}}(L)\text{-}\mathbf{S}$ will be, indeed, a functor. The property of $LH_{\mathcal{Q}}$ being a full embedding is easily seen from the definition of $LH_{\mathcal{Q}}$.

5 Relations Between L-$\mathcal{Q}\mathbf{S}$ and $\mathcal{Q}\mathbf{P}^{op}$

In this section, with regard to the adjunction $AT \dashv A\Psi : \mathbf{P}^{op} \to A\text{-}\mathbf{S}$, we will establish an adjunction between L-$\mathcal{Q}\mathbf{S}$ and $\mathcal{Q}\mathbf{P}^{op}$, and then restrict it to an equivalence by introducing suitable definitions of spatiality in $\mathcal{Q}\mathbf{P}$ and sobriety in L-$\mathcal{Q}\mathbf{S}$. We begin with some preliminaries.

Proposition 5. *[5] For categories $\boldsymbol{A}$, $\boldsymbol{B}$, a full subcategory $\boldsymbol{A}'$ of $\boldsymbol{A}$ and a full subcategory $\boldsymbol{B}'$ of $\boldsymbol{B}$, if $F \dashv G : \boldsymbol{A} \to \boldsymbol{B}$ is an adjoint pair of functors with the property that for all $X \in Ob(\mathbf{A}')$ and for all $Y \in Ob(\mathbf{B}')$,*

$$G(X) \in Ob(\mathbf{B}') \text{ and } F(Y) \in Ob(\mathbf{A}') \tag{1}$$

then the restriction F' of F to $\boldsymbol{B}'$ and the restriction G' of G to $\boldsymbol{A}'$ form an adjoint pair of functors $F' \dashv G' : \boldsymbol{A}' \to \boldsymbol{B}'$.

Corollary 2. *[5] Under the considerations in Proposition 5, if $G : \boldsymbol{A} \to \boldsymbol{B}$ and $F : \boldsymbol{B} \to \boldsymbol{A}$ are equivalences inverse to each other and satisfy (1), then $G' : \boldsymbol{A}' \to \boldsymbol{B}'$ and $F' : \boldsymbol{B}' \to \boldsymbol{A}'$ are equivalences inverse to each other.*

Lemma 3. *Let $\boldsymbol{P}(\mathcal{Q})$ and $\boldsymbol{S}(L,\mathcal{Q})$ stand for the image of $\mathcal{Q}\boldsymbol{P}$ under $G_{\mathcal{Q}}$ and the image of L-$\mathcal{Q}\boldsymbol{S}$ under $LH_{\mathcal{Q}}$, respectively. Then, the functor $G_{\mathcal{Q}}(L)T : G_{\mathcal{Q}}(L)\text{-}\boldsymbol{S} \to \boldsymbol{P}^{op}$ assigns to each $\boldsymbol{S}(L,\mathcal{Q})$-object a $\boldsymbol{P}(\mathcal{Q})$-object.*

Proof. Since each $\mathbf{S}(L,\mathcal{Q})$-object W possesses the property that $\mathfrak{O}_{G_{\mathcal{Q}}(L)}(W)$ is a $(\mathcal{Z}_1,\mathcal{Z}_2)$-complete poset and $G_{\mathcal{Q}}(L)T(W) = G_{\mathcal{Q}}(\mathfrak{O}_{G_{\mathcal{Q}}(L)}(W))$, it is clear from the definition of $\mathbf{P}(\mathcal{Q})$ that $G_{\mathcal{Q}}(L)T(W)$ is a $\mathbf{P}(\mathcal{Q})$-object.

Lemma 4. *Let $\mathcal{Z}_1$ and $\mathcal{Z}_2$ preserve surjectivity. Then, the functor $G_{(\mathcal{Z}_1,\mathcal{Z}_2)}(L)\Psi$ maps each $\boldsymbol{P}(\mathcal{Z}_1,\mathcal{Z}_2)$-object to an $\boldsymbol{S}(L,\mathcal{Z}_1,\mathcal{Z}_2)$-object.*

Proof. The proof can be done by using analogous arguments to those in the proof of [5, Lemma 6], so it is omitted here.

Lemma 5. *Let $\boldsymbol{P}(L,\mathcal{Q})_s$ denote the full subcategory of $\boldsymbol{P}(\mathcal{Q})$ of all $G_{\mathcal{Q}}(L)$-spatial objects. Then $G_{\mathcal{Q}}(L)\Psi$ maps each $\boldsymbol{P}(L,\mathcal{Q})_s$-object to an $\boldsymbol{S}(L,\mathcal{Q})$-object.*

Proof. The required result can be verified in a similar fashion to [5, Lemma 8].

Definition 5. *Let P be a $(\mathcal{Z}_1,\mathcal{Z}_2)$-complete poset, and (X,τ) an L-$\mathcal{Q}$-space.*
(i) P is L-$\mathcal{Q}$-spatial iff $G_{\mathcal{Q}}(P)$ is $G_{\mathcal{Q}}(L)$-spatial.
(ii) (X,τ) is L-$\mathcal{Q}$-sober iff $LH_{\mathcal{Q}}(X,\tau)$ is $G_{\mathcal{Q}}(L)$-sober.

Theorem 3. *Let L-$\mathcal{Q}\boldsymbol{P}_s$ be the full subcategory of $\mathcal{Q}\boldsymbol{P}$ of all L-$\mathcal{Q}$-spatial objects, and L-$\mathcal{Q}\boldsymbol{S}_s$ be the full subcategory of L-$\mathcal{Q}\boldsymbol{S}$ of all L-$\mathcal{Q}$-sober objects. Then the following statements are true:*
(i) There is a pair of adjoint functors $LT_{\mathcal{Q}} \dashv L\Psi_{\mathcal{Q}} : L\text{-}\mathcal{Q}\boldsymbol{P}_s^{op} \to L\text{-}\mathcal{Q}\boldsymbol{S}$.
(ii) L-$\mathcal{Q}\boldsymbol{P}_s$ is dually equivalent to L-$\mathcal{Q}\boldsymbol{S}_s$.
(iii) If $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are surjectivity-preserving, then there is a pair of adjoint functors $LT_{(\mathcal{Z}_1,\mathcal{Z}_2)} \dashv L\Psi_{(\mathcal{Z}_1,\mathcal{Z}_2)} : (\mathcal{Z}_1,\mathcal{Z}_2)\boldsymbol{P}^{op} \to L\text{-}(\mathcal{Z}_1,\mathcal{Z}_2)\boldsymbol{S}$.

Proof. (i) To prove the assertion, we apply Proposition 5, and choose $F \dashv G : \mathbf{A} \to \mathbf{B}$ therein as $G_{\mathcal{Q}}(L)T \dashv G_{\mathcal{Q}}(L)\Psi : \mathbf{P}^{op} \to G_{\mathcal{Q}}(L)\text{-}\mathbf{S}$. By considering the fact that the co-domain of $G_{\mathcal{Q}}(L)T$ lies in $\mathbf{P}(L,\mathcal{Q})_s^{op}$, and using Lemma 3 and Lemma 5, we observe that the condition (1) in Proposition 5 is satisfied for $\mathbf{A}' = \mathbf{P}(L,\mathcal{Q})_s^{op}$ and $\mathbf{B}' = \mathbf{S}(L,\mathcal{Q})$. Thus the adjunction in question follows from Proposition 5 and the fact that $\mathbf{P}(L,\mathcal{Q})_s$ and $\mathbf{S}(L,\mathcal{Q})$ are, respectively, isomorphic to L-$\mathcal{Q}\mathbf{P}_s$ and L-$\mathcal{Q}\mathbf{S}$.

(ii) is an application of Corollary 2 to the equivalences $G_{\mathcal{Q}}(L)\Psi_s : G_{\mathcal{Q}}(L)\text{-}\mathbf{SpaP}^{op} \to G_{\mathcal{Q}}(L)\text{-}\mathbf{SobS}$ and $G_{\mathcal{Q}}(L)T_s : G_{\mathcal{Q}}(L)\text{-}\mathbf{SobS} \to G_{\mathcal{Q}}(L)\text{-}\mathbf{SpaP}^{op}$. By making use of the isomorphism between $\mathbf{P}(L,\mathcal{Q})_s$ and L-$\mathcal{Q}\mathbf{P}_s$, the full subcategory of $\mathbf{S}(L,\mathcal{Q})$ of all $G_{\mathcal{Q}}(L)$-sober spaces and L-$\mathcal{Q}\mathbf{S}_s$, the required equivalence follows from Lemma 3 and Lemma 5.

(iii) Firstly, pick $F \dashv G : \mathbf{A} \to \mathbf{B}$ as $G_{(\mathcal{Z}_1,\mathcal{Z}_2)}(L)T \dashv G_{(\mathcal{Z}_1,\mathcal{Z}_2)}(L)\Psi : \mathbf{P}^{op} \to G_{(\mathcal{Z}_1,\mathcal{Z}_2)}(L)\text{-}\mathbf{S}$, $\mathbf{A}' = \mathbf{P}(\mathcal{Z}_1,\mathcal{Z}_2)^{op}$ and $\mathbf{B}' = \mathbf{S}(L,\mathcal{Z}_1,\mathcal{Z}_2)$ in Proposition 5. Then, since $(\mathcal{Z}_1,\mathcal{Z}_2)\mathbf{P}$ and L-$(\mathcal{Z}_1,\mathcal{Z}_2)\mathbf{S}$ are, respectively, isomorphic to $\mathbf{P}(\mathcal{Z}_1,\mathcal{Z}_2)$ and $\mathbf{S}(L,\mathcal{Z}_1,\mathcal{Z}_2)$, we obtain the questioned pair of adjoint functors from Proposition 5 by making use of Lemma 3 and Lemma 4.

Theorem 3 is an extension of the main result of [5] ([5, Theorem 2]) to the poset-valued $\mathcal{Q}$-spaces, and has direct applications to many familiar categories of ordered-structures. We gather some (but not all) of them in the following final result.

Corollary 3. *Let L be an object of **Pos** (**Blatt**, $\boldsymbol{SUP}^{\wedge}$, **MCPos**, $\boldsymbol{INF}^{\vee}$, $\boldsymbol{INF}^{\uparrow}$, **CLat**, σ**ComLat** and **QF**). Then, there are dual adjunctions between **Pos** and L-**BS**, **Blatt** and L-**BlatBS**, $\boldsymbol{SUP}^{\wedge}$ and L-**Top**, **MCPos** and L-**CSp**, $\boldsymbol{INF}^{\vee}$ and L-**TCSp**, L-$\boldsymbol{INF}_s^{\uparrow}$ and L-**ACSp**, **CLat** and L-**ATSp**, σ**ComLat** and L-**Alex**, L-$\boldsymbol{QF}_s$ and L-**PreTop**. Furthermore, there are dual equivalences*

*between L-**Pos**$_s$ and L-**BS**$_s$, L-**Blatt**$_s$ and L-**BlatBS**$_s$, L-**SUP**$_s^{\wedge}$ and L-**Top**$_s$, L-**MCPos**$_s$ and L-**CSp**$_s$, L-**INF**$_s^{\vee}$ and L-**TCSp**$_s$, L-**INF**$_s^{\uparrow}$ and L-**ACSp**$_s$, L-**CLat**$_s$ and L-**ATSp**$_s$, L-σ**ComLat**$_s$ and L-**Alex**$_s$, L-**QF**$_s$ and L-**PreTop**$_s$.*

References

1. Abramsky, S., Jung, A.: Domain theory. In: Abramsky, S., Gabbay, D., Maibaum, T.S.E. (eds.) Handbook of logic in computer science. vol. 3. , pp. 1–168. The Clarendon Press, Oxford University Press, New York (1994)
2. Adámek, J., Herrlich, H., Strecker, G.E.: Abstract and Concrete Categories. Wiley, New York (1990)
3. Adámek, J., Nelson, E., Reiterman, J.: Continuous algebras revisited. J. Comput. Syst. Sci. 51, 460-471 (1995)
4. Banaschewski, B., Bruns, G.: The fundamental duality of partially ordered sets. Order 5, 61-74 (1988)
5. Demirci, M: $(\mathcal{Z}_1, \mathcal{Z}_2)$-complete partially ordered sets and their representations by $\mathcal{Q}$-spaces. Appl. Categ. Struc. doi: 10.1007/s10485-012-9277-4.
6. Erné, M: Bigeneration in complete lattices and principle separation in posets. Order 8, 197-221 (1991)
7. Erné, M: The ABC of order and topology. In: Herrlich, H., Porst, H. E. (eds.) Category Theory at Work, pp.57-83. Heldermann, Berlin, Germany (1991)
8. Erné, M: Algebraic ordered sets and their generalizations. In: Rosenberg, I., Sabidussi, G. (eds.) Algebras and Orders, Proc. Montreal, 1992, pp. 113-192, Kluwer Academic Publishers, Amsterdam, Netherlands (1993)
9. Erné, M: $\mathcal{Z}$-continuous posets and their topological manifestation, Appl. Categ. Structures 7, 31-70 (1999)
10. Erné, M: General Stone duality. Topology and its Applications 137, 125-158 (2004)
11. Erné, M.: Choiceless, pointless, but not useless: dualities for preframes. Appl. Categ. Structures 15, 541-572 (2007)
12. Erné, M.: Closure. In : Mynard, F., et al. (eds.) Beyond topology. Contemporary Mathematics 486, pp. 163-238. American Mathematical Society, Providence (2009)
13. Gierz, G., Hofmann, K.H., Keimel K., Lawson, J.D., Mislove, M., Scott, D.S.: Continuous Lattices and Domains. Cambridge University Press, Cambridge (2003)
14. Goguen, J. A.: Categories of V -sets. Bull. Amer. Math. Soc. 75, 622-624 (1969)
15. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer Academic Publishers, Dordrecht, Boston, London (1998)
16. Markowsky, G.: Categories of chain-complete posets. Theor. Comput. Sci. 4, 125-135 (1977)
17. Nelson, E.: Z-continuous algebras. In : Banaschewski, B., Hoffmann, R.E. (eds.) Continuous Lattices. Proc. Conf., Bremen 1979. Lect. Notes Math. 871, pp. 315-334. Springer-Verlag, Berlin (1981)
18. Novak, D.: Generalization of continuous posets. Trans. Amer. Math. Soc. 272, 645-667 (1982)
19. Powers, R.C., Riedel, T.: Z-Semicontinuous posets. Order 20, 365-371 (2003)
20. Rodabaugh, S. E.: Point-set lattice-theoretic topology. Fuzzy Sets and Systems 40, 297–345 (1991)
21. Wright, J. B., Wagner, E. G., Thatcher, J. W.: A uniform approach to inductive posets and inductive closure. Theoret. Comput. Sci. 7, 57-77 (1978)
22. Zadeh, L.A.: Fuzzy sets. Inform. and Control 8, 338-353 (1965)

RNA pseudoknot prediction through stochastic conjunctive grammars

Ryan Zier-Vogel[1] and Michael Domaratzki[2⋆]

[1] Department of Computer Science
Memorial University of Newfoundland
St. John's, NL A1B 3X5 Canada
rzvogel@gmail.com
[2] Department of Computer Science
University of Manitoba
Winnipeg, MB R3T 2N2 Canada
mdomarat@cs.umanitoba.ca

Abstract. We present an extension of conjunctive grammars to allow stochastic parsing. By extending the CYK parsing algorithm and applying labelled training data, we use these stochastic conjunctive grammars to predict RNA secondary structures that include pseudoknots. We design specific grammars to predict H-type pseudoknots and obtain 74 % sensitivity and 89 % specificity with a grammar with 133 non-terminals and 552 productions in binary normal form.

1 Introduction

The secondary structure of a single strand of RNA is formed when the nucleotides in the strand bind together and form base pairs, which causes the RNA strand to fold on itself. This folded structure is referred to as the secondary structure of the RNA strand. An example of RNA secondary structure is seen in Figure 1. The secondary structure of RNA is important for several types of RNA strands, including ribosomal RNA and RNA viruses [3, 20].

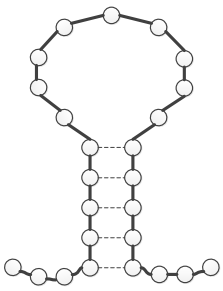


Fig. 1: This structure is known as a hairpin. The thick black line is the backbone of the RNA and the dotted lines are the bonds in the secondary structure.

In this paper, we are interested in the prediction of secondary structure of RNA which includes pseudoknots. A pseudoknot is a non-nested secondary structure, where bonds form between bases in positions i and k, as well as between positions j and ℓ, where $i < j < k < \ell$. See Figure 2 for an example of a type of

⋆ Research supported in part by a grant from NSERC.

pseudoknot. Predicting RNA secondary structure, given only the linear sequence of nucleotides of the strand, is a well-studied problem in bioinformatics. Prediction of RNA secondary structure including pseudoknots has been shown to be NP-hard [1]. However, by restricting allowable solutions to secondary structures that do not contain pseudoknots, the computational complexity of RNA secondary structure prediction can be accomplished in cubic time [12, 24].

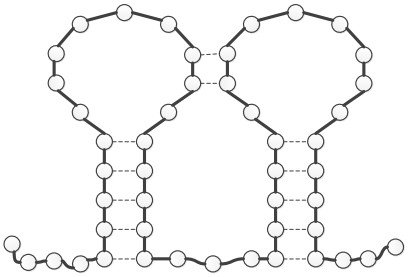


Fig. 2: This pseudoknotted structure is known as a kissing hairpin. This structure represents a pseudoknot as the bonds between the two loops form a non-nested structure.

In this paper, we use grammars to predict the pseudoknotted secondary structure of RNA. Grammars work by rewriting non-terminals symbols using a set of productions. By assigning probabilities to each production, stochastic grammars can be trained to predict the most probable structure for an RNA sequence. We develop and train stochastic grammars to solve the problem, which is a form of machine learning [2] previously used for prediction of RNA secondary structure without pseudoknots and other problems in bioinformatics [7, 18, 4, 19, 9]. We also note that different grammatical models have been pursued for pseudoknot RNA secondary structure prediction [22, 17, 5, 13, 11, 10, 16]. Some of these models are more complex than what we propose here, while others lack a stochastic element which allows them to be employed as machine learning tools.

The grammars we develop in this paper are stochastic versions of conjunctive grammars [14, 15]. We obtain fast, accurate results for our grammars, including a 133 non-terminal, 552 production stochastic conjunctive grammar able to predict H-type pseudoknots with 74 % sensitivity and 89 % specificity. The H-type pseudoknot is the most common type of pseudoknot and is made up of a combination of two hairpins as seen in Figure 3.

2 Stochastic Conjunctive Grammars

While stochastic context-free grammars have been successfully employed to predict pseudoknot-free secondary structure [18, 6], a more powerful grammar is needed to predict pseudoknotted secondary structure. We introduce a class of grammars called Stochastic Conjunctive Grammars (SCGs). A SCG G is a five-tuple $G = (\Sigma, N, P, \Phi, S)$ where

- Σ is the set of terminal symbols.
- N is the set of non-terminal symbols.
- P is the set of productions for the grammar. The productions are written as $A \to \alpha_1 \ \& \ \ldots \ \& \ \alpha_n$ where $n \geq 1$ and $\alpha_i \in (\Sigma \cup N)^*$,
- $\Phi : P \to [0, 1]$. Φ associates a probability to each production. For each $A \in N$, let $P_A \subseteq P$ be the set of all productions with A on the left-hand side; then Φ must satisfy $\sum_{r \in P_A} \Phi(r) = 1$ for all $A \in N$.
- $S \in N$ is the start symbol.

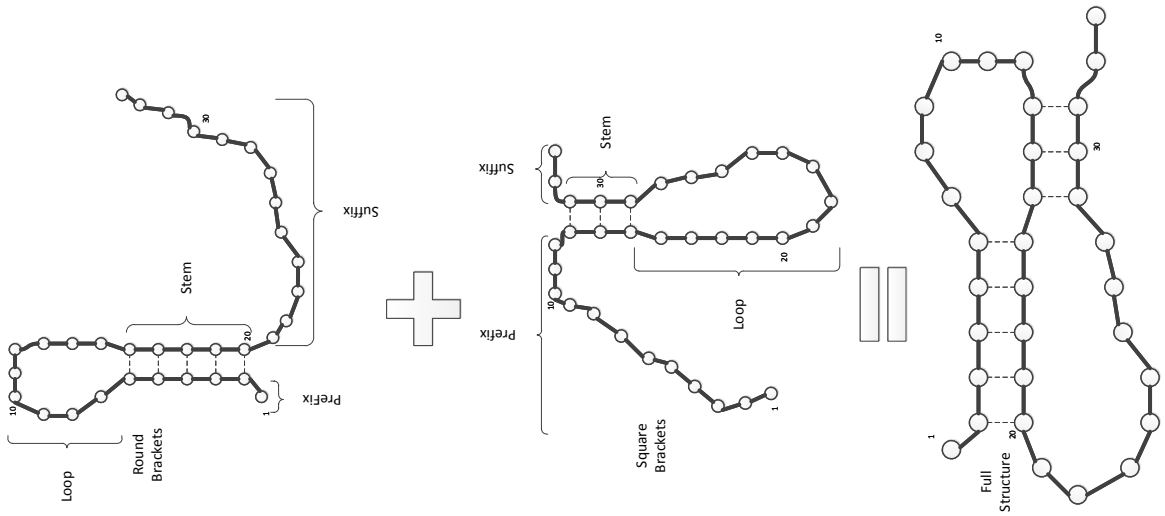


Fig. 3: An example of how a H-type pseudoknot is a combination of two hairpins.

SCGs are an extension of conjunctive grammars (CGs), introduced by Okhotin [14]. For a recent survey of conjunctive grammars and the related class of Boolean grammars, see Okhotin [15]. Esik and Kuich [8] previously considered fuzzy conjunctive and Boolean grammars, which are similar to the stochastic version of conjunctive grammars. The work of Brown and Wilson [4], which considers intersections of stochastic context-free grammars, is a particular case of our work. We note that the current formulation allows for recursive definitions involving conjunction, which is not possible with a finite intersection of context-free grammars.

The & symbol denotes conjunction or intersection which is a unique component of CGs and SCGs. In SCGs, $A \to \alpha_1 \& \ldots \& \alpha_n$ means A is rewritten by $(\alpha_1 \& \ldots \& \alpha_n)$ and that all derivations by α_i must lead to the same terminal word or the entire derivation is unsuccessful. If at any point two conjuncts being intersected are not the same sequence of terminal symbols then the whole derivation is invalid; see Okhotin [15] for more details. In addition to the derivation process, in an SCG, the probability associated with a production must be used to calculate the current probability of the entire derivation.

Consider the following SCG.

- $\Sigma = \{b, c\}$
- $N = \{S, A, B, C\}$
- $S \in N$ is the start symbol.

The productions of the SCG, along with the probability associated with each, are

$$\begin{aligned}
&S \to A \& BC \ (p = 1)\\
&A \to cA \ (p = .45)\\
&A \to bA \ (p = .45)\\
&A \to \varepsilon \ (p = .1)\\
&B \to bB \ (p = .75)\\
&B \to \varepsilon \ (p = .25)\\
&C \to cB \ (p = .75)\\
&C \to \varepsilon \ (p = .25)
\end{aligned} \tag{1}$$

Consider a derivation with this grammar. The current probability of the sentential form is specified after each step of the derivation.

$$
\begin{aligned}
&S \\
&\Rightarrow A\&BC \ (p = 1) \\
&\Rightarrow bA\&BC \ (p = .45) \\
&\Rightarrow bbA\&BC \ (p = .2025) \\
&\Rightarrow bbbA\&BC \ (p = .091125) \\
&\Rightarrow bbbcA\&BC \ (p = .04100625) \\
&\Rightarrow bbbccA\&BC \ (p = .01845281) \\
&\Rightarrow bbbcc\&BC \ (p = .00184528) \\
&\Rightarrow bbbcc\&bBC \ (p = .00138396) \\
&\Rightarrow bbbcc\&bbBC \ (p = .00103797) \\
&\Rightarrow bbbcc\&bbbBC \ (p = .00077848) \\
&\Rightarrow bbbcc\&bbbC \ (p = .00019462) \\
&\Rightarrow bbbcc\&bbbcC \ (p = .00014596) \\
&\Rightarrow bbbcc\&bbbccC \ (p = .00010947) \\
&\Rightarrow bbbcc\&bbbcc \ (p = .00002737) \\
&\Rightarrow bbbcc \ (p = .00002737)
\end{aligned}
\tag{2}
$$

Whenever a non-terminal is rewritten, the probability of the derivation is multiplied by the probability of the new production being used. Since a probability is assigned to the whole production, when simplifying the & symbol the probability is not affected (i.e., when going from the second last to last step the probability remains at .00002737). The final sequence is *bbbcc* and the probability of the generating this sequence with this derivation is .00002737. Note that other derivations of this same sequence may also have nonzero probabilities. To deal with the issue of multiplying probabilities, which can lead to underflow, log odds are used: the logarithms of the probabilities are used and are summed instead of being multiplied. Log odds are negative numbers, where a lower value corresponds to a smaller probability.

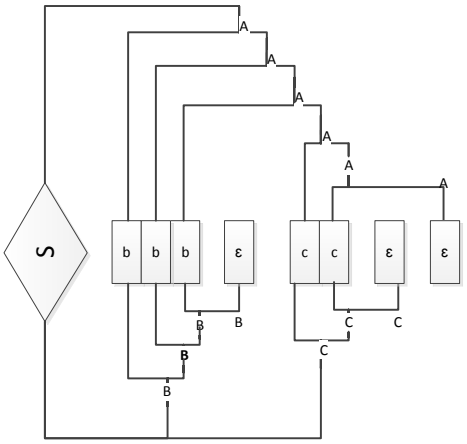


Fig. 4: This parse tree is a representation of the derivation in Equation 2, with probabilities omitted. The children of the diamond shaped node will each represent a different conjunct.

Parse trees can be used to represent CG or SCG derivations (an example of a CG parse tree is seen in Figure 4). The only difference between a stochastic parse tree and a non-stochastic parse tree is that there is a probability assigned to the tree once the parse is finished. For a CG parse tree there are some additional rules that are added so that the parse trees can handle more than one conjunct. It is important when there is more than one conjunct that the leaf nodes are used in the same order for each conjunct. If the leaf nodes are not used in the same order it would be a representation of an invalid derivation.

2.1 Parsing Algorithm

We give a version of the standard CYK algorithm for SCGs. Previously, Okhotin [14] has given a version of the CYK algorithm for conjunctive grammars. As usual, the version of the CYK algorithm for SCGs will only work on grammars that are in binary normal form; a grammar is in binary normal form if all productions have the form of $A \to B_1C_1\&B_2C_2\&\ldots\&B_mC_m$ or $A \to a$. All CGs are able to be transformed into binary normal form as shown by Okhotin [14]. Since the productions of SCGs are the same as CGs, the transformation will work for SCGs.

We have adapted the CYK algorithm for conjunctive grammars [14] to SCGs. Let $x_1 \cdots x_n$ be the input sequence. Equation (3) gives the recurrence for $\gamma(i, j, v)$, which is the most likely probability of any parse of $v \Rightarrow^* x_i \cdots x_j$. Equation (4) gives the set required information to trace the most probable parse. For more information on the CYK algorithm derived from these recurrences, see Zier-Vogel [23]. As we have designed our grammars by hand, without probabilities, we are not required to convert SCGs to a binary normal form; training will then assign probabilities to these productions.

$$\gamma(i,j,v) = \max_{v \to \alpha \in P_v} \sum_{v \to BC \in \alpha} \max_{1 \le k \le j-1} \{\gamma(i,k,B) + \gamma(k+1,j,C) + \Phi(v \to BC)\} \tag{3}$$

$$\tau(i,j,v) = \left\{ \bigcup_{v \to BC \in \alpha} (B,C,k) | \gamma(i,j,v) = \sum_{v \to BC \in \alpha} \max_{1 \le k \le j-1} \{\gamma(i,k,B) + \gamma(k+1,j,C) + \Phi(v \to BC)\} \right\} \tag{4}$$

3 Training Method

We tested our grammars using all the RNA pseudoknotted sequences in the pseudoBase++ database [21]. The database has 304 pseudoknotted RNA sequences with known secondary structures. However, we focused on predicting H-type pseudoknots, the most common type of pseudoknot, with 236 out of 304 RNA sequences in pseudoBase++ having only this type of pseudoknot. These grammars were designed based on statistics about 235 of the 236 H-type pseudoknots (one sequence was excluded because it did not fit the standard pattern for a H-type pseudoknot). The statistics that were gathered were parts of a hairpin structure. We called these parts prefix, stem, loop and suffix (an example can be seen in Figure 3). These grammars yielded much better results than a general grammar to predict RNA pseudoknotted secondary structure.

All pseudoknot-prediction SCGs were trained to assign probabilities to the individual productions. For all productions with a nonterminal A on the left-hand side, the probability assigned to a particular production $A \to \alpha_1\& \cdots \&\alpha_n$ is the count of the number of times that this production is used in any training data divided by the total number of times that productions with A on the left-hand side are used in any training data. To calculate when a production is used, labelled training data was required, which gives a sequence-structure pair. An example of labelled training data can be seen in Figure 5.

The algorithms for training will take a sequence and its structure and split it into two parts (an example is seen in Figure 6). The importance of splitting the sequence and structure into both the round and square brackets representing the non-nested pseudoknot structure is due to the grammars predicting the square and round brackets independently through separate conjuncts in the SCG. For more information on training of probabilities in the SCG, please see Zier-Vogel [23].

```
UAGGGGGGUCAGGGUCAGGAGCCCCCCCCUGAACCCAGGAUAACCCUCACUGUCGGGGGGCA
::::::::::((((((((:[[[[[[[)))):))))::::::::::::::::::]]]]]]]:
```

Fig. 5: Sample of labelled training data, from the pseudoBase++ database [21], sequence id PKB78. The top sequence is the RNA strand and the sequence at the bottom represents its structure. A pair of brackets represents a base pair in the structure. Different types of brackets are needed to denote non-nested structures. The colons represent unpaired nucleotides.

```
UAGGGGGGUCAGGGUCAGGAGCCCCCCCCUGAACCCAGGAUAACCCUCACUGUCGGGGGGCA
::::::::::(((((((:::::::::))))):))))::::::::::::::::::::::::::

UAGGGGGGUCAGGGUCAGGAGCCCCCCCCUGAACCCAGGAUAACCCUCACUGUCGGGGGGCA
::::::::::::::::::::[[[[[[[:::::::::::::::::::::::::::]]]]]]]:
```

Fig. 6: The RNA sequence in Figure 5, split into two. The top two lines are the sequence and structure only for the round brackets and the other uses the square brackets.

4 Grammars

We considered three distinct grammars for RNA secondary structure prediction with pseudoknots. Each of these grammars was designed based on our desired structures: the productions were given to reflect the idealized secondary structure, and then converted to binary normal form. Training then assigned probabilities to the productions.

4.1 Secondary Structure Prediction with Grammars

An example of a CG parse tree for RNA secondary structure can be seen in Figure 7. We associate the secondary structure of a parsed RNA strand as follows: for any production of the form $X \to a_1 Y a_2$, where a_1, a_2 are nucleotides, we consider a_1 and a_2 bonded in the secondary structure. Note how the top tree in Figure 7 only generates stems on the round brackets and the bottom tree generates the square brackets. When both of these structures are combined it will generate an H-type pseudoknot. Since the structure is pseudoknotted it will have non-nested brackets; these non-nested brackets will be represented by different bracket types, either a round bracket or a square bracket. Note how the two trees in Figure 7 generate stems in distinct spots.

In order to predict the secondary structure of a sequence, it is provided as input to the CYK algorithm for SCGs. The most likely parse structure is then interpreted as a secondary structure, as above.

4.2 General pseudoknot grammar

The first grammar that we developed was designed to predict any type of pseudoknots in a strand. The grammar was designed with productions for separately predicting two distinct nested structures, which were then interleaved (through conjunction) to give a non-nested pseudoknot structure.

The sensitivity of this grammar was on average 0.248 and specificity was 0.8. The structures often predicted by this grammar would have the round and square brackets in the same position, which is counted as a misprediction, leading to the low sensitivity. The specificity was high because the positions with the double bracket prediction did align with either a square or round bracket, which means the grammar rarely predicted a bracket where there was not one. As seen in Figure 8 both set of brackets are aligned with the square brackets in the actual structure.

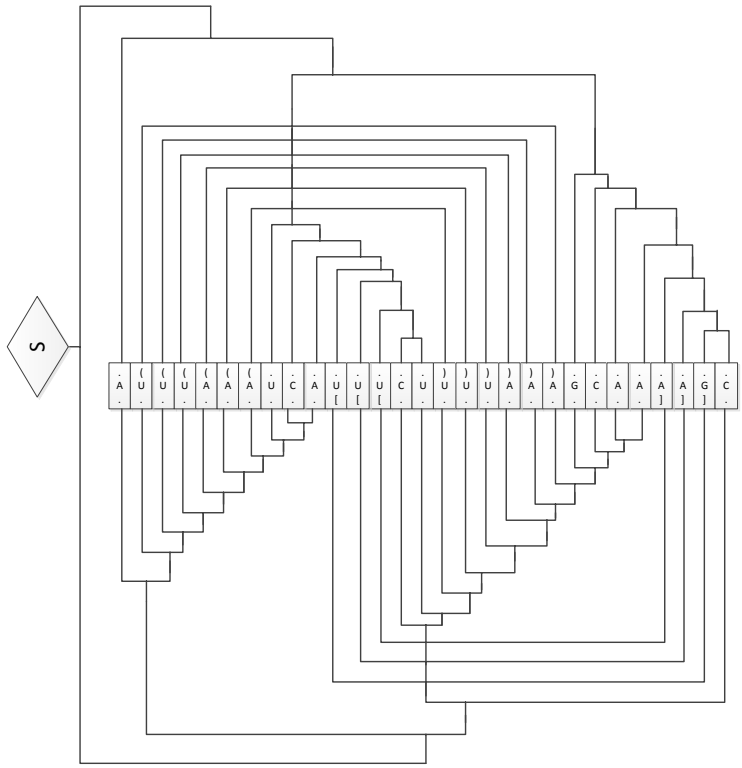


Fig. 7: This parse tree will start at S. The tree above and below the sequence will generate distinct nested structures. Internal nodes have labels, corresponding to nonterminals, which have been removed for clarity.

```
Sequence          GGGGUGCGACUCCCCCGUCUAUCCUGAACGUCAUCAGGACCA
Actual            ..............(((...[[[[[[)))...]]]]]]...
Predicted round   ::::::::::::::::::::(((((::::::)))))):::
Predicted square  ::::::::::::::::::::[[[[[[::::::]]]]]]:::
Misses                          xxx   xxxxxxxxx   xxxxxx
```

Fig. 8: Sample of output of a prediction by the first grammar, sequence id PKB13 on pseudoBase++. The top sequence is the RNA strand, the next sequence is to the actual structure of the pseudoknot. The third sequence is how the algorithm predicted the round brackets and the last sequence is the algorithm's prediction of the square brackets.

4.3 H-type grammar

To solve the problem of double bracket prediction, we developed a grammar that, unlike the first grammar, would be built to only predict one type of pseudoknot, called an H-type pseudoknot (see Figure 3 for an example of an H-type pseudoknot). As the H-type is the most common type (77% of pseudoknots in pseudoBase++ are H-type), we chose to construct a grammar which can predict H-type pseudoknots exclusively. Out of the 236 H-type pseudoknots we excluded one (sequence and structure given in Figure 9); all other H-type pseudoknots will follow a pattern of open round brackets, then open square brackets, then close round brackets and finally close square brackets.

```
CCUCCCGGGAGA~ACUGCCUGAUAGGGUGCUUGCGAGUGCCCCGGGAGGUCUCGUAG
((((((((((.~.))(((......)))((.[[[[[[.))))))))))).]]]]]].
```

Fig. 9: Excluded H-type pseudoknot. Its structure differs significantly from other H-types (ID PKB181).

The next grammar was made to model the structure of the H-type pseudoknot. Counts were gathered about the structure of H-type pseudoknots. Since an H-type is constructed from two interconnected hairpins the structural elements that were counted are prefix, suffix and stem from each hairpin. An example of how two hairpins make a H-type and what is meant by prefix, suffix and stem are seen in Figure 3.

All 235 RNA sequences were used to give information on prefix, suffix, stem and loop length. The statistics that were collected were minimums, maximums, averages and in general the counts of how many sequences are of a certain length. These statistics were collected for the four structures prefix, stem, loop and suffix, as well as for the sequences themselves.

We incorporated this information into a new grammar with 117 non-terminals and 436 productions in binary normal form. In particular, the minimum length of prefixes, suffixes and stems were incorporated into chains of productions to enforce these minimum lengths. For details on the grammar, see Zier-Vogel [23].

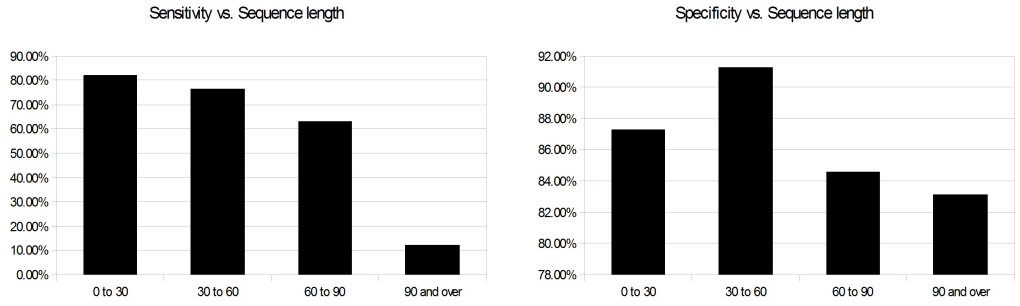
A sample of 117 sequences were chosen at random and the training algorithm was run on those sequences. Then the trained grammar was tested on the remaining 118 sequences. These steps were done 100 times and then the average sensitivity and specificity across all 100 iterations was calculated. The average sensitivity was 0.743 and the average specificity of 0.895.

4.4 Improved H-type grammar

The previous grammar omitted information on the loop structure, i.e., the sequence of unpaired nucleotides that is formed when base pairs form a stem. When the information about loop length is added, the grammar will increase the number of non-terminals to 133 and the number of productions to 552; see Zier-Vogel [23] for the complete grammar. A sample of 117 sequences were chosen at random and the training algorithm was run for those sequences. Then the trained grammar was tested on the remaining 118 sequences. These step were done 100 times and then the average sensitivity and specificity across all 100 iterations was calculated. The average sensitivity was 0.754 and the average specificity of 0.891.

This last grammar also has on average 25 sequences out of 117 that are predicted 100% correctly. The H-type prediction grammar without loop information will get an average of 18 sequences 100% correct and the first grammar will get an average of 3 sequences 100% correct.

The last grammar for predicting H-type pseudoknots also has a much higher sensitivity when it is predicting sequences that have length less than 60, after which point the sensitivity drops off. This is not seen for specificity which is consistently high independently of the length of sequences. These statistics can be seen in Figure 10. Since the average length of a H-type sequence is about 40, the lowered sensitivity after length 60 does not affect too many of the sequences.



(a) Sensitivity

(b) Specificity

Fig. 10: (a) A break down of average sensitivity for 100 tests based on various length sequences. (b) A break down of an average specificity for 100 tests based on various length sequences.

5 Discussion

Our prediction algorithm relies on the CYK algorithm, which has a time complexity of $O(n^3m)$ and space complexity of $O(n^2m)$ where n is the length of a sequence and m is the number of non-terminals in the grammar. Previous grammar based predictors, for example, the pair stochastic tree adjoining grammar designed by [13], had more intensive run-times. For instance, the pair stochastic tree adjoining algorithm has a running time of $O(n^5)$.

Even though the model of Matsui *et al.* [13] has a higher sensitivity, our algorithm performs our predictions faster. Also, the model of Matsui *et al.* [13] does not have the versatility of being able to predict all H-types sequences but can only make predictions based on phylogenetic family.

6 Conclusions and Future Work

In this paper we extended the class of CGs to SCGs. This extension allows SCGs to be used for machine learning in bioinformatics. We also provided a generalized CYK algorithm for SCGs.

We also designed grammars that were able to predict RNA pseudoknotted secondary structure. We were unable to build a grammar that was able to predict all types of pseudoknots successfully. However, when we focused the predictions to only H-type pseudoknots, the grammars were very successful with an average sensitivity of over 75% and an average specificity of over 89%. It may be difficult to extend our grammar to other pseudoknot types as there are few training examples. Currently, pseudoBase++ lists just over 70 non-H-type pseudoknotted sequences.

As well, it would be interesting to find other kinds of problems that this class of grammars can be applied to in bioinformatics, including possibly RNA-RNA interaction prediction, which was previously considered by Kato *et al.* [10].

References

1. T. Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104(1–3):45–62, 2000.
2. P. Bildi and S. Brunak. *Bioinformatics: The Machine Learning Approach.* The MIT Press, 2001.
3. Ian Brierley, Simon Pennell, and Robert J. C. Gilbert. Viral RNA pseudoknots: versatile motifs in gene expression and replication. *Nature Reviews Microbiology*, 5:598–610, August 2007.
4. M. Brown and C. Wilson. RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. In *Pacific Symposium on Biocomputing*, pages 109–125. 1996.

5. L. Cai, R. Malmberg, and Y. Wu. Stochastic model of RNA pseudoknotted structures: a grammatical approach. *Bioinformatics*, 19(1):66–73, 2003.
6. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.
7. S. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22(11):2079–2088, 1994.
8. Z. Esik and W. Kuich. Boolean fuzzy sets. *International Journal of Foundations of Computer Science*, 18(6):1197–1207, 2007.
9. X.-Y. Fang, Z.-G. Luo, and Z.-H. Wang. Predicting RNA secondary structure using profile stochastic context-free grammars and phylogenic analysis. *Journal of Computer Science and Technology*, 23(4):582–589, 2008.
10. Y. Kato, T. Akutsu, and H. Seki. A grammatical approach to RNA–RNA interaction prediction. *Pattern Recognition*, 42(4):531–538, 2006.
11. Y. Kato, H. Seki, and T. Kasami. RNA pseudoknotted structure prediction using stochastic multiple context-free grammar. *IPSJ Digital Courier*, 2:655–664, 2006.
12. D. Mathews, S. Schroeder, D. Turner, and M. Zuker. Predicting RNA secondary structure. In R. Gesteland, T. Cech, and J. Atkins, editors, *The RNA World*. Cold Spring Harbor Laboratory Press, 2006.
13. H. Matsui, K. Sato, and Y. Sakakibara. Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures. *Bioinformatics*, 21(11):2611–2617, 2005.
14. A. Okhotin. Conjunctive grammars. *Journal of Automata, Languages and Combinatorics*, 6(4):519–535, 2001.
15. A. Okhotin. *Conjunctive and Boolean grammars: the true general case of the context-free grammars*. Preprint available at `http://users.utu.fi/aleokh/papers/boolean_survey.pdf`, 2012.
16. Sanguthevar Rajasekaran, Sahar Al Seesi, and Reda A. Ammar. Improved algorithms for parsing ESLTAGs: A grammatical model suitable for RNA pseudoknots. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(4):619–627, 2010.
17. E. Rivas and S. Eddy. The language of RNA: a formal grammar that includes pseudoknots. *Bioinformatics*, 16(4):334–340, 2000.
18. Y. Sakakibara, M. Brown, R. Underwood, I. Mian, and D. Haussler. Stochastic context-free grammars for modeling RNA. *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences: Biotechnology Computing*, 5:284–293, 1994.
19. Yasubumi Sakakibara. Grammatical inference in bioinformatics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1051–1062, 2005.
20. D. Staple and S. Butcher. Pseudoknots: RNA structures with diverse functions. *PLoS Biology*, 3:e213, 2005.
21. M. Taufer, A. Licon, R. Araiza, D. Mireles, F. H. D. van Batenburg, A. Gultyaev, and M.-Y. Leung. Pseudobase++: an extension of pseudobase for easy searching, formatting and visualization of pseudoknots. *Nucleic Acids Research*, 37:D127–D134, 2009.
22. Yasuo Uemura, Aki Hasegawa, Satoshi Kobayashi, and Takashi Yokomori. Tree adjoining grammars for RNA structure prediction. *Theoretical Computer Science*, 210:277–303, 1999.
23. R. Zier-Vogel. Predicting RNA secondary structure using a stochastic conjunctive grammar. Master's thesis, University of Manitoba, `http://hdl.handle.net/1993/8453`, 2012.
24. M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, 9:133–148, 1981.

A Kripke model for subrecursion

Joaquín Díaz Boils[1] and José P. Úbeda[2]

[1] Departament de Matemàtiques
Universitat Jaume I de Castelló
boils@uji.es
[2] Departament de Lògica y Filosofia de la Ciència
Universitat de València
jose.p.ubeda@uv.es

Abstract. A new point of view of Symmetric Monoidal Comprehensions is introduced to give an analogous interpretation about how some subrecursive function classes can be categorically constructed. It is given, in particular, a general setting to perform subrecursive hierarchies like the known as *Grzegorzcyk Hierarchy*. That is done by constructing a category intended to emulate safe recursion and composition.

Keywords: Symmetric monoidal Comprehension, Grzegorzcyk Hierarchy, Safe recursion, Applicative structure, Kripke model.

1 Introduction

J. Otto in [10] suggests a close relation between Kripke structures and composition of functions in the sense of S. J. Bellantoni and J. Cook [1]. He gives a characterization of the first three levels of Grzegorzcyk Hierarchy by using symmetric monoidal 2- and 3-Comprehensions.

Based on results of M. Wirz [11] and S. J. Bellantoni and S.J. Niggl [2] and generalizing concepts of J. Otto, we give in [4] a categorical counterpart of generalized *safe composition* and *safe recursion*. A new categorical setting was defined in order to characterize the subrecursive classes belonging to (the whole) *Grzegorzcyk Hierarchy*. This was achieved by means of *coercion functors* over a symmetric monoidal category endowed with certain recursion schemes that imitate the known as *bounded recursion scheme*.

We summarize those concepts in Section 2 and remite the reader to [4] for details. In that section a diferent point of view of subrecursion, and of *Grzegorzcyk Hierarchy* in particular, is given. By considering Kripke applicative structures we introduce in Section 3 a general setting for subrecursion which is specialized (as a Kripke model) to the Grzegorzcyk Hierarchy in Section 4. It can be of interest because of the fact that Kripke models are more readable for non experts in Category Theory and because they have been developed more extensively than categorical comprehensions. In section 5 we show how a cartesian category that contains a general instance of safe recursion and safe composition in categorical

terms can be defined. It is called in that section a safe semantic category. Section 6 exposes briefly a summary of the results presented and future research directions starting from it.

2 Categorical characterization of Grzegorzcyk hierarchy

Definition 1. *We denote by* $\mathbf{n}$ *the (symmetric monoidal) category whose set of objects is* $n = \{0, 1, \ldots, n-1\}$ *and whose arrows, denoted by* $m_{i,j}$ *with* $0 \leq i \leq j < n$*, are those unique* $m_{i,j} : i \longrightarrow j$*.*

Definition 2. *Let be for* $0 \leq k < n-1$ *functors* $id : \mathbf{n} \longrightarrow \mathbf{n}$*,* $T_k : \mathbf{n} \longrightarrow \mathbf{n}$ *and* $G_k : \mathbf{n} \longrightarrow \mathbf{n}$ *such that for all* $j \in \mathbf{n}$*:*

$$id(j) = j \qquad T_k(j) = \begin{cases} j+1 & \text{if } j = k \\ j & \text{if } j \neq k \end{cases} \qquad G_k(j) = \begin{cases} j-1 & \text{if } j = k+1 \\ j & \text{if } j \neq k+1 \end{cases}$$

We will refer to T_k *and* G_k *as* coercion functors*.*

Definition 3. *Let* $\epsilon_k : G_k \Longrightarrow id$ *and* $\eta_k : id \Longrightarrow T_k$ *(*$0 \leq k \leq n-2$*) be such that*

$$\epsilon_k(i) = \begin{cases} m_{i,i} & \text{if } i \neq k+1 \\ m_{i-1,i} & \text{if } i = k+1 \end{cases} \qquad \eta_k(i) = \begin{cases} m_{i,i} & \text{if } i \neq k \\ m_{i,i+1} & \text{if } i = k \end{cases}$$

for $i = 0, ..., n-1$*.*

We say that, given a symmetric monoidal category (**SM** category in the sequel[3]) $\mathcal{C} = (C, \otimes, \top, a, l, \sigma)$, a tuple $(\mathcal{C}, \langle T_k^{\mathcal{C}} \rangle, \langle G_k^{\mathcal{C}} \rangle, \langle \eta_k^{\mathcal{C}} \rangle, \langle \epsilon_k^{\mathcal{C}} \rangle)$ is a *model* of $(\mathbf{n}, \langle T_k \rangle, \langle G_k \rangle, \langle \eta_k \rangle, \langle \epsilon_k \rangle)$ if $T_k^{\mathcal{C}}$, $G_k^{\mathcal{C}}$, $\eta_k^{\mathcal{C}}$ and $\epsilon_k^{\mathcal{C}}$ satisfy the same commutative diagrams than T_k, G_k, η_k and ϵ_k $(0 \leq k < n-1)$.

Definition 4. *A* ***SM*** n-Comprehension *is a tuple* $(\mathcal{C}, \langle T_k^{\mathcal{C}} \rangle, \langle G_k^{\mathcal{C}} \rangle, \langle \eta_k^{\mathcal{C}} \rangle, \langle \epsilon_k^{\mathcal{C}} \rangle)$ *(*$0 \leq k < n-1$*) where*

1. $\mathcal{C}$ *is a* ***SM*** *category;*
2. *for every* k *such that* $0 \leq k < n-1$ *the functors* $T_k^{\mathcal{C}}, G_k^{\mathcal{C}} : \mathcal{C} \longrightarrow \mathcal{C}$ *are* ***SM*** *functors;*
3. *for every* k *such that* $0 \leq k < n-1$ *the natural transformations* $\eta_k^{\mathcal{C}} : id \Longrightarrow T_k^{\mathcal{C}}$ *and* $\epsilon_k^{\mathcal{C}} : G_k^{\mathcal{C}} \Longrightarrow id$ *are* ***SM*** *transformations;*
4. $(\mathcal{C}, \langle T_k^{\mathcal{C}} \rangle, \langle G_k^{\mathcal{C}} \rangle, \langle \eta_k^{\mathcal{C}} \rangle, \langle \epsilon_k^{\mathcal{C}} \rangle)$ *is a model of* $(\mathbf{n}, \langle T_k \rangle, \langle G_k \rangle, \langle \eta_k \rangle, \langle \epsilon_k \rangle)$*.*

Definition 5. *We define* doctrines $\mathcal{A}^n$ *as those whose objects are* ***SM*** n*-Comprehensions* $(\mathcal{C}, \langle T_k \rangle, \langle G_k \rangle, \langle \eta_k \rangle, \langle \epsilon_k \rangle)$ *with* $0 \leq k \leq n-2$

[3] Concepts are taken from [6] and [10].

– *endowed with a diagram (named* initial diagram*)* $\top \xrightarrow{0} N_0 \xrightarrow{s} N_0$ *in* $\mathcal{C}$ *such that*

$$T_0(\top \xrightarrow{0} N_0 \xrightarrow{s} N_0) = \top \xrightarrow{1_\top} \top \xrightarrow{1_\top} \top$$

where, if we define recursively for each $i = 1, 2, ..., n-2$ *the objects* N_i *by*

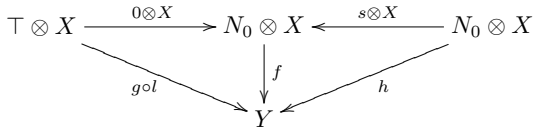
$$N_1 = G_0 N_0 \qquad N_{i+1} = G_i N_i$$

we have for each $i = 0, 1, ..., n-2$ *and* $j = 0, 1, ..., n-1$

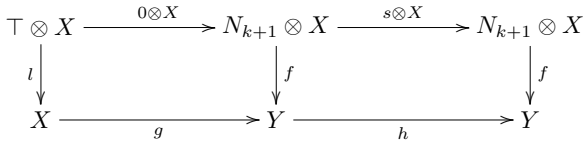
$$T_i N_j = \begin{cases} \top & \text{if } i = j = 0 \\ N_{i-1} & \text{if } i = j \neq 0 \\ N_j & \text{other} \end{cases} \qquad G_i N_j = \begin{cases} N_{i+1} & \text{if } i = j \\ N_j & \text{other} \end{cases}$$

– *closed under*

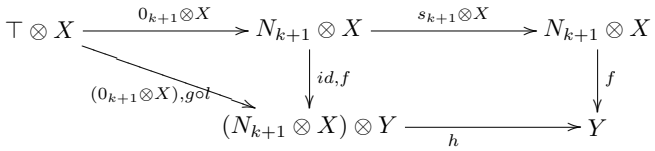
- flat recursion*: for all* $g : X \longrightarrow Y$ *and* $h : N_0 \otimes X \longrightarrow Y$ *where* $T_0 X$ *and* $T_0 Y$ *are isomorphic to* $\top$ *there exists a unique morphism* $f : N_0 \otimes X \longrightarrow Y$ *in* $\mathcal{C}$ *such that the following diagram commutes*



- safe ramified recursion *on each level* k*: for all* $k = 0, 1, ..., n-2$ *and for all morphisms* $g : X \longrightarrow Y$ *and* $h : Y \longrightarrow Y$ *where* $T_{k+1}...T_0 Y$ *is isomorphic to* $\top$ *there exists a unique* $f : N_{k+1} \otimes X \longrightarrow Y$ *in* $\mathcal{C}$ *such that the following diagram commutes*



– *for all cartesian object*[4] *in* n-Comprehensions $(\mathcal{C}, \langle T_k^{\mathcal{C}} \rangle, \langle G_k^{\mathcal{C}} \rangle, \langle \eta_k^{\mathcal{C}} \rangle, \langle \epsilon_k^{\mathcal{C}} \rangle)$ *it is also closed under* safe dependent recursion *in each level* k*: for all* $k = 0, 1, ..., n-2$ *and all morphisms* $g : X \longrightarrow Y$ *and* $h : (N_{k+1} \otimes X) \otimes Y \longrightarrow Y$ *where* $T_k...T_0 Y$ *is isomorphic to* $\top$ *and* X *and* Y *are cartesian objects there exists a unique* $f : N_{k+1} \otimes X \longrightarrow Y$ *in* $\mathcal{C}$ *such that the following diagram commutes*



[4] We call *cartesian objects* in $\mathcal{A}^n$ the objects in the form $\bigotimes_{i=0}^{n-1} N_i^{\alpha_i}$.

Example 1. We consider *Set* as a **SM** category (actually a *cartesian* **SM** with the tensor product given by $(\times, 1)$).

- let be the sets $\mathbb{N}_k = \{(k,n)/n \in \mathbb{N}\}$ with $k = 0, 1, ..., n-1$;
- we denote every pair (k,n) by n_k;
- let be $s_k(n_k) = (n+1)_k$ for $k = 0, ..., n-2$ and $n_k \in \mathbb{N}_k$ the *successor function in* $\mathbb{N}_k$;
- let T_k be the endofunctor in *Set* defined by

$$T_k X = \begin{cases} \emptyset & \text{if } k = 0 \text{ and } X = (0,n) \text{ with } n \in \mathbb{N} \\ 1 & \text{if } k = 0 \text{ and } X = \mathbb{N}_0 \\ (k-1,n) & \text{if } k \neq 0 \text{ and } X = (k,n) \text{ with } n \in \mathbb{N} \\ \mathbb{N}_{k-1} & \text{if } k \neq 0 \text{ and } X = \mathbb{N}_k \\ T_k Y \otimes T_k Z & \text{if } X = Y \otimes Z \\ X & \text{other} \end{cases}$$

- analogously we can define an endofunctor G_k by

$$G_k X = \begin{cases} (k+1,n) & \text{if } X = (k,n) \text{ with } n \in \mathbb{N} \\ \mathbb{N}_{k+1} & \text{if } X = \mathbb{N}_k \\ G_k Y \otimes G_k Z & \text{if } X = Y \otimes Z \\ X & \text{other} \end{cases}$$

- let $\eta_k : G_k \Rightarrow id$ and $\epsilon_k : id \Rightarrow T_k$ be natural transformations;
- an *initial diagram* in *Set* given by $1 \xrightarrow{0} \mathbb{N}_0 \xrightarrow{s} \mathbb{N}_0$;

According to the Definition 5 we have all the equations making

$$(Set, \langle T_k \rangle, \langle G_k \rangle, \langle \eta_k \rangle, \langle \epsilon_k \rangle)$$

a **SM** n-Comprehension.

Proposition 1. *The $\mathcal{A}^n$ doctrine is endowed with an initial **SM** n-*Comprehension

$$(\mathcal{J}^n, \langle T_k \rangle, \langle G_k \rangle, \langle \eta_k \rangle, \langle \epsilon_k \rangle).$$

We abbreviate it as $\mathcal{J}^n$.

Definition 6. *Let $\Gamma : \mathcal{J}^n \longrightarrow Set$ be defined by $\Gamma X = \mathcal{J}^n(\top, X)$ and $\Gamma f = f \circ -$.*[5]

Proposition 2. *The image of the objects N_k by the functor Γ are sets whose elements have the form $\Gamma N_k = \{std_k n / n \in \mathbb{N}\}$ where $std_k : \mathbb{N} \longrightarrow \Gamma N_k$ is defined by the scheme*

$$\begin{cases} std_k 0 = 0_k \\ std_k(sn) = s_k(std_k n) \end{cases} \qquad \text{with } k = 0, 1, ..., n-1$$

[5] This is a special case of the know as *global sections functor*.

Corollary 1. *$\Gamma N_k = \mathbb{N}_k$ for all $k = 0, 1, ..., n-1$.*

Definition 7. *The binary functions sequence F_j for $j \in \mathbb{N}$ is recursively defined as:*

$$F_0(x,y) = y+1$$
$$F_1(x,y) = x+y$$
$$F_2(x,y) = (x+1)(y+1)$$

and the scheme

$$\begin{cases} F_{n+1}(0,y) = F_n(y+1, y+1) \\ F_{n+1}(x+1,y) = F_{n+1}(x, Fn+1(x,y)) \end{cases}$$

for $n > 1$. ([5, p. 28])

Definition 8. *We will say that a function class X is closed respect to the operation of* bounded recursion *if, given three functions $g, h, j \in X$, every function f satisfying the three following conditions*

$$\begin{cases} f(u,0) = g(u) \\ f(u,x+1) = h(u,x,f(u,x)) \\ f(u,x) \leq j(u,x) \end{cases}$$

belongs to X. ([5, p. 14])

Definition 9. *The* Grzegorzcyk Hierarchy *is the sequence of classes $\mathcal{E}^n$ of numeric functions, where $\mathcal{E}^n$ is the smallest class such that:*

1. *it includes successor, first and second projections and $F_n(x,y)$ as initial functions and*
2. *is closed under the operations of substitution and bounded recursion. ([5, p. 29])*

We denote by $\mathcal{K}^n$ the initial doctrine $\mathcal{J}^n$ with the difference that we don't allow any action of a functor T_k over a morphism of it. We also denote by $\mathcal{G}^n$ the functions set $\{\Gamma f : \mathbb{N}_{n-1}^{\alpha} \longrightarrow \mathbb{N}_{n-1} / \alpha \in \mathbb{N}, f \in \mathcal{K}^n\}$.

The following Theorem is the main result of [4, p. 97].

Theorem 1. *[Categorical Characterization of* Grzegorzcyk Hierarchy*]*

$$\mathcal{E}^n = \mathcal{G}^n \text{ for all } n \geq 3$$

3 Kripke structures for Grzegorzcyk Hierarchy

The Kripke applicative structures defined by J. Mitchell and E. Moggi [9] generalize the structures defined by S. Kripke in [7] for the intuitionistic logic.

Definition 10. *A* Kripke applicative structure *$\mathcal{A}$ is a tuple*

$$\left\langle (W, \leq), \{A_w^{\sigma}\}, \{App_w^{\sigma,\tau}\}, \{i_{w,w'}^{\sigma}\} \right\rangle$$

where

– $(W, \leq)$ *is a partially ordered set of* possible worlds;[6]
– $\{A^\sigma_w\}$ *is a family of sets indexed by types and worlds* $w \in W$*;*
– $\{App^{\sigma,\tau}_w\}$ *is a family of* application functions $App^{\sigma,\tau}_w : A^{\sigma\to\tau}_w \times A^\sigma_w \longrightarrow A^\tau_w$ *indexed by pairs of types* σ *and* τ *and worlds* $w \in W$*;*
– $\{i^\sigma_{w,w'}\}$ *is a family of* transition functions $i^\sigma_{w,w'} : A^\sigma_w \longrightarrow A^\sigma_{w'}$ *indexed by types* σ *and worlds* $w \leq w'$*.*

Under the following conditions:

1. *the transition function* $i^\sigma_{w,w} : A^\sigma_w \longrightarrow A^\sigma_w$ *is the identity;*
2. *transition functions compose in the following way* $i^\sigma_{w',w''} \circ i^\sigma_{w,w'} = i^\sigma_{w,w''}$ *for each worlds* $w \leq w' \leq w''$*;*[7]
3. *application and transition functions commute in the following sense*

$$i^\tau_{w,w'}(App^{\sigma,\tau}_w(f,a)) = App^{\sigma,\tau}_{w'}((i^{\sigma\to\tau}_{w,w'}f),(i^\sigma_{w,w'}a))$$

for each $f \in A^{\sigma\to\tau}_w$ *and for each* $a \in A^\sigma_w$ *what can be expressed by the following commutative diagram*

$$\begin{array}{ccc} A^{\sigma\to\tau}_{w'} \times A^\sigma_{w'} & \xrightarrow{App^{\sigma,\tau}_{w'}} & A^\tau_{w'} \\ \uparrow{\scriptstyle i^{\sigma\to\tau}_{w,w'}\times i^\sigma} & & \uparrow{\scriptstyle i^\tau_{w,w'}} \\ A^{\sigma\to\tau}_w \times A^\sigma_w & \xrightarrow[App^{\sigma,\tau}_w]{} & A^\tau_w \end{array}$$

An analogous definition of a Kripke applicative structure could be given by taking directly a cartesian category where sets A^σ_w are the (indexed) objects over a partial ordered set seen itself as a category with a unique arrow between every pair of objects expressing the partial order. In that case, which we don't develope here, *App* would be (indexed) arrows of the former category. Related to the indexing point of view we have the following concepts.

From a Kripke applicative structure $\mathcal{A}$ we can define for all type σ a functor

$$\phi_\sigma : W \longrightarrow Set$$

given by $\phi_\sigma(w) = A^\sigma_w$ and $\phi_\sigma(l_{w,w'}) = i^\sigma_{w,w'}$ for each $l_{w,w'} : w \longrightarrow w'$ such that $w \leq w'$. The conditions making a functor out of ϕ_σ for each type σ are satisfied by conditions 1. and 2. in the definition of Kripke applicative structure.

Definition 11. *A* global element *a of type* σ *in a Kripke applicative structure* $\mathcal{A}$ *is an assignation* $w \mapsto a_w$ *from worlds to elements such that* $a_w \in A^\sigma_w$ *and when* $w \leq w'$ *then* $i^\sigma_{w,w'}a_w = a_{w'}$*.*

[6] It can be seen as a category whose objects are the elements of W and its morphisms $l_{w,w'}$ for all $w, w' \in W$ such that $w \leq w'$.

[7] From this we know that there is exactly one transition function from A^σ_w to $A^\sigma_{w'}$ for all $w \leq w'$.

Proposition 3. *Every global element a of type $\sigma \to \tau$ induces a natural transformation from ϕ_σ to ϕ_τ given by $App_w^{\sigma,\tau}(a_w, \cdot)$.*

Definition 12. *We will say that a Kripke applicative structure is* extensional *if for every $f, g \in A_w^{\sigma\to\tau}$ such that $(i_{w,w'}^{\sigma\to\tau} f)(a) = (i_{w,w'}^{\sigma\to\tau} g)(a)$ with $w' \geq w$ and $a \in A_w^\sigma$ we have $f = g$.*

Definition 13. *A Kripke applicative structure is a* Kripke model *if it is extensional and there are no empty types.*

4 The Grzegorzcyk Hierarchy by a Kripke model

From the doctrine structure $\mathcal{A}^n$ given in Definition 5 it can be observed that what we are actually constructing is a Kripke (model) structure. It is developed in the following.

The type system we will make use of, and that we will denote as $\mathcal{ST}$ because of its similarity with known as *Simple Type Theory*, is specified as follows:

Definition 14. *Let be a* type system *consisting of:*

1. *natural numbers α as* 1-types *where*
   (a) *if $\alpha = 0$ we have $N_k^0 = \top$;*
   (b) *if $\alpha = 1$ we have sets $N_k^1 = N_k$;*
   (c) *if $\alpha = j + 1$ we have sets $N_k^{j+1} = N_k \otimes N_k^j$, para $0 < j$.*
2. function types *are those in the form $\alpha \to \beta$ where α and β are 1-types giving rise to an object $N_k^{\alpha\to\beta}$ subset of $(N_k^\beta)^{N_k^\alpha}$.*

We are now going to give our main exemple of the functions a Kripke applicative structure like ours can give rise to. If we take $W = \mathbf{n}$ as defined above, we can construct a Kripke applicative structure.

$$\left\langle \mathbf{n}, \{N_k^\alpha\}, \{App_k^{\alpha,\beta}\}, \{i_{k,j}^\alpha\} \right\rangle$$

which we will denote by $\mathcal{REC}_n$.

We now give a description of some elements belonging to the (set-theoretic point of view of) $\mathcal{K}^n$ have an interpretation in our Kripke applicative structure $\mathcal{REC}_n$:

objects: objects are $\mathbb{N}_k^\alpha$ with 1-type α and world k where worlds correspond to the different levels;[8]

functional objects: objects in the form $\mathbb{N}_k^{\alpha\to\beta}$ where α and β are 1-types giving rise to a subset of $(\mathbb{N}_k^\beta)^{\mathbb{N}_k^\alpha}$, el conjunto of the morfisms $f : \mathbb{N}_k^\alpha \longrightarrow \mathbb{N}_k^\beta$ in $\mathcal{K}^n$;

transition functions: transition functions $i_{m,j}^1 : \mathbb{N}_m \longrightarrow \mathbb{N}_j$ are the composite functors $G_{j-1}...G_m$ in the **SM** n-Comprehension $(\mathcal{J}^n, \langle T_k \rangle, \langle G_k \rangle, \langle \eta_k \rangle, \langle \epsilon_k \rangle)$ for worlds $m \leq j \leq n-1$. We then also have for α a 1-type, $i_{m,j}^\alpha$ as the functor $i_{m,j}^1$ applied to N_m^α, remember that functors G_k preserve $\otimes$;

[8] Those defined in [4] as *levels of some object* N, relating to the work of D. Leivant in [8], to an arbitrary tensor power α; they correspond to the Definition 5.

1-type global elements: a 1-type global element α in $\mathcal{REC}_n$ is an assignation ϕ_α from $\mathbf{n}$ to $\mathcal{REC}_n$ such that $\phi_\alpha(0)$ is a vector with α components belonging to $\mathbb{N}_0^\alpha$ and for $0 < k < n-1$ we have[9]

$$\phi_\alpha(k+1) = i_{k,k+1}^{\alpha}(\phi_\alpha(k));$$

functional global elements: a function type global element $\alpha \to \beta$ in $\mathcal{REC}_n$ is an assignation $\phi_{\alpha\to\beta}$ from $\mathbf{n}$ to $\mathcal{REC}_n$ such that $\phi_{\alpha\to\beta}(0)$ is a function in the set $(\mathbb{N}_0^\beta)^{\mathbb{N}_0^\alpha}$ and for $0 < k < n-1$ we have[10]

$$\phi_{\alpha\to\beta}(k+1) = i_{k,k+1}^{\alpha\to\beta}(\phi_{\alpha\to\beta}(k));$$

general transitions: functors $i_{k,j}^{\alpha\to\beta}$ with $k \leq j$ in $(\mathcal{J}^n, \langle T_k\rangle, \langle G_k\rangle, \langle \eta_k\rangle, \langle \epsilon_k\rangle)$ send functions in the form $f_k : \mathbb{N}_k^\alpha \longrightarrow \mathbb{N}_k^\beta$ to functions in the form[11]

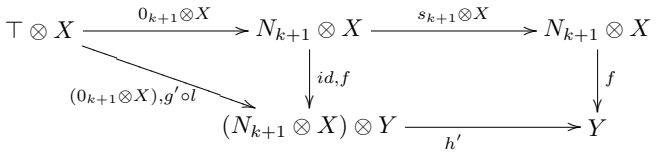
$$f_j : \mathbb{N}_j^\alpha \longrightarrow \mathbb{N}_j^\beta;$$

extensionality: $\mathcal{REC}_n$ is extensional since for all $f, g \in \mathbb{N}_k^{\alpha\to\beta}$ such that

$$(i_{k,j}^{\alpha\to\beta} f)(n) = (i_{k,j}^{\alpha\to\beta} g)(n)$$

with $n-1 \geq j \geq k$ and $n \in N_k^\alpha$ we have necessarily $f = g$. This is because what we get in $\mathbb{N}_j^\alpha$ are copies of the functions in $\mathbb{N}_k^\alpha$ making grow the argument levels as it is explained in [4];

recursive operator: every $\mathbb{N}_{k+1}^{\alpha\to\beta}$ with α and β 1-types contains, for every $X = \mathbb{N}_{k+1}^\alpha$, $Y = \mathbb{N}_{k+1}^\beta$, $g \in N_k^{\alpha\to\beta}$ and $h \in \mathbb{N}_k^{(\alpha+\beta+1)\to\beta}$, a unique $f \in \mathbb{N}_{k+1}^{1+\alpha\to\beta}$ such that the following diagram commutes



where $g' = i_{k,k+1}^{\alpha\to\beta}(g)$ and $h' = i_{k,k+1}^{(\alpha+\beta+1)\to\beta}(h)$.

It is important to note here that the chosen of the type theory from which we build up the applicative structure is crucial for the kind of functions we get. For exemple, consider that the above recursive diagram for $\mathcal{REC}_n$ could not be defined from the types available in $\mathcal{ST}$.

We see that $\mathcal{REC}_n$ contains all functions belonging to $\mathcal{E}^0, \ldots, \mathcal{E}^{n-1}$ in the *Grzegorzcyk Hierarchy* and moreover $\bigcup_{n\in\mathbb{N}} \mathcal{REC}_n$ will then contain all primitive recursive functions.

[9] In our case, and considering the initial **SM** n-Comprehension, each vector in N_0^α gives rise to a global element.

[10] Functions belonging to the first *Grzegorzcyk Hierarchy* class $\mathcal{E}^0$ are function type global elements and they also live in all successive $\mathcal{E}^n$ for $n > 0$.

[11] f_j is obtained by appliying the functor $i_{k,j}^1$ to the function f_k.

Proposition 4. *$\mathcal{RELC}_n$ is a Kripke model for every $n \in \mathbb{N}$.*

Definition 15. *Let $\phi_\alpha : \mathbf{n} \longrightarrow Set$ be the functor given by $\phi_\alpha(k) = \mathbb{N}_k^\alpha$ and $\phi_\alpha(m_{k,j}) = i_{k,j}^\alpha$ for every $m_{k,j} : k \longrightarrow j$ such that $k, j \in \mathbf{n}$ with $k \leq j$.*

5 Safe semantic categories

Instead of J. Mitchell and E. Moggi [9] given two types α and β such that $\alpha \neq \beta$ we can ensure that the sets in $\mathcal{RELC}_n$ in the form N_k^α and N_k^β with $k \in \mathbf{n}$ belonging to the initial **SM** n-Comprehension $\mathcal{J}^n$ are different (although isomorphic). In this way the cartesian closed category generated by the Kripke applicative structure $\mathcal{RELC}_n$, that we will denote by $\mathcal{C}_{\mathcal{RELC}_n}$ in the following definition, can be constructed from its presheaves and not uniquely from its types (see [9, page 15]).

Definition 16. *Let $\mathcal{C}_{\mathcal{RELC}_n}$ be the category whose objects are functors ϕ_α in $Set^{\mathbf{n}}$ from Definition 15 for each $\alpha \in \mathbb{N}$ and whose morphisms are natural transformations between those functors.*

In this way we are considering a subcategory of $Set^{\mathbf{n}}$ where the objects in $\mathcal{C}_{\mathcal{RELC}_n}$ generate all sets in $\mathcal{RELC}_n$. Natural transformations in the form $\phi_\alpha \longrightarrow \phi_\beta$ are generated from every arrow $m_{k,j} : k \longrightarrow j$ in $\mathbf{n}$ by commutative squares

$$\begin{array}{ccc} \phi_\alpha(k) & \longrightarrow & \phi_\beta(k) \\ {\scriptstyle \phi_\alpha(m_{k,j})}\downarrow & & \downarrow{\scriptstyle \phi_\beta(m_{k,j})} \\ \phi_\alpha(j) & \longrightarrow & \phi_\beta(j) \end{array} \tag{1}$$

As pointed out in [9], $\mathcal{C}_{\mathcal{RELC}_n}$ is cartesian if $\mathcal{RELC}_n$ has products and a terminal object and for every natural number α there exists a unique functor φ_α in $\mathcal{C}_{\mathcal{RELC}_n}$.

Particularly, and looking beyond $\mathcal{J}^n$, in $\mathcal{C}_{\mathcal{RELC}_n}$ it is formalized a way to perform safe recursion and safe composition: its objects are copies of powers of $\mathbb{N}$, it can be endowed with a certain *safe recursive operator* as the one given in Section 4 and it is endowed with coercion functors $\varphi_\alpha(m_{k,j})$.

From Proposition 3 we observe that every global element f of type $\alpha \to \beta$ induces a natural transformation in the form

$$App^{\alpha,\beta}(f, \cdot) : \phi_\alpha \longrightarrow \phi_\beta$$

giving $App_k^{\alpha,\beta}(f, N_k^\alpha)$ for a set N_k^α in $\mathcal{RELC}_n$ where we supose k to be the domain and codomain world of f. That is, by fixing the first variable in *App* we are producing assignations in the form $N_k^\alpha \longrightarrow N_k^\beta$ for every world k in $\mathbf{n}$ and f having functional type $\alpha \to \beta$ in the world k. Natural components of $App^{\alpha,\beta}(f, \cdot)$ are then arrows in Set in the form

$$App_k^{\alpha,\beta}(f, N_k^\alpha) : \phi_\alpha(k) \longrightarrow \phi_\beta(k)$$

and therefore arrows $\mathbb{N}_k^\alpha \longrightarrow \mathbb{N}_k^\beta$.

The naturality condition for $App_k^{\alpha,\beta}(f,\cdot)$ expresses safe composition. That is due to the fact that in it an expression of every morphism $f \in N_j^{\alpha\to\beta}$ in $\mathcal{RELC}_n$ is obtained in terms of other morphisms where the variables belong to the world j. An output belonging to a world j, in general, does not depend on inputs belonging to worlds lower than j.

That is, elements and concepts in $\mathcal{C}_{\mathcal{RELC}_n}$ gather the semantics of safety and therefore it can be viewed as a *safe semantic category* for a certain Kripke applicative structures.

6 Conclusion and future work

The content of this talk is the initial point of a further study to get a full description of subrecursive function classes from Kripke models. For that purpose one could get rid of comprehension structures as defined here to work just with a Kripke applicative structure endowed with certain recursion schemes. It would make extensive use of the categorical concept of presheaf. This line is currently being developed.

References

1. Bellantoni, S. J., Cook, S.: New Recursion-Theoretic Characterization of the Polytime Functions. Comput. Complexity 2, 97–110 (1992)
2. Bellantoni, S. J., Niggl, K.: Ranking Primitive Recursions: the Low Grzegorczyk Classes Revisited. SIAM J. Comput. 29(2), 401–415 (1999)
3. Clote, P.: Computation models and function algebras. In: Griffor, E.R. (ed) Handbook of Computability Theory, pp. 589-681. Elsevier Science B.V. (1999).
4. Díaz Boils, J.: Comprensiones Categoriales y Subrecursión. PhD Thesis. Universitat de València (2012) Available at: `http://roderic.uv.es/handle/10550/24423`
5. Grzegorczyk, A.: Some classes of recursive functions. Rozprawy matematyczne 4, 1–45, (1953)
6. Kelly, G. M.: Basic Concepts of Enriched Category Theory. Reprints in Theory and Applications of Categories 10, (2005)
7. Kripke, S.: Semantical Analysis of Intuitionistic Logic I. In: Dummett, M., J. N. Crossley, J. N. (eds) Formal Systems and Recursive Functions, pp. 92–129. North Holland (1965)
8. Leivant, D.: Ramified recurrence and computational complexity I: Word algebras and polytime. In: Clote, P., Remmel, J. (eds.) Feasible Mathematics II, pp. 320–343. Birkhause (1994)
9. Mitchell, J., Moggi, E.: Kripke-style models for typed lambda calculus. Journal of Pure and Applied Algebra 51, 99–124 (1991) Available at: `www.disi.unige.it/person/MoggiE/ftp/kripke.pdf`
10. Otto, J.: Complexity Doctrines. Ph.D. thesis, McGill University (1995)
11. Wirz, M.: Characterizing the Grzegorczyk hierarchy by safe recursion. CiteSeer, (1999) `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.3374.`

How to reduce backtracking in propositional Intuitionistic logic

Guido Fiorino

Dipartimento di Informatica Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, P.zza dell'Ateneo Nuovo 1, 20126 Milano, Italy

Abstract. We discuss a technique to reduce search space in propositional Intuitionistic logic. Our technique is a syntactical criterion based on the sign property and can be used by any kind of calculus.

1 Introduction

In this paper we continue our investigations to speed-up proof search in propositional Intuitionistic logic (**Int**). In [3] we have introduced a calculus that advances the well-known calculus of [8], in [5] we have identified conditions allowing one to replace propositional variables with logical constants and an implementation of these achievements is the prover fCube [4]. In the cited papers the focus is on the rules or on the logical apparatus as a whole and the strategy using the calculus is not considered. In this note we introduce a criterion applicable to any strategy to reduce the non-determinism in **Int** proof search. To sake of concreteness, we present our results applied to a specific tableau calculus. However, it will be clear that our results can be applied to any tableau or sequent calculus.

Every step of a deduction has multiple choices and if the proof search fails, then every step is a point where to backtrack because the order of application of the rules is relevant, that is, not permutable. However, there is a special kind of rules that does not require to backtrack, because if a rule of this kind is used and a proof is not found, then no proof exists. This kind of rules is called *invertible*: the lack of a proof for the conclusion of an invertible rule implies the lack of a proof for the premise. The order of application of invertible rules is irrelevant: different permutations of their application always brings to the same result. On the other hand, the *non-invertible* rules require to backtrack: the lack of a proof for the conclusion of a non-invertible rule does not imply the lack of a proof for the premise. The application of non-invertible rules is not permutable, because different permutations of their applications produce different results. Thus, if in a step of a deduction a non-invertible rule $\mathcal{R}$ is applied and subsequenty to this step no proof is not found, then a different application of $\mathcal{R}$ or the application of a different rule could bring to find a proof. Note that such application is useless if we already know that it does not bring to find a proof.

Thus, it is useful to find strategies to reduce as much as possible the application of non-invertible rules, because this shrinks the search space for a proof. The aim of this paper is to present a criterion to get this goal. This work can be

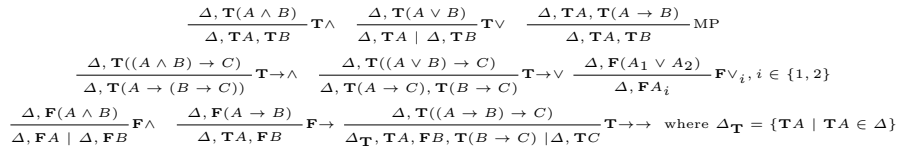


Fig. 1. The calculus

seen as an extension of [2] where the syntax of sequent calculus LJQ gives rise to a strategy allowing to avoid some backtracking steps. The criterion we present is strictly related to our previous investigations in [5]. For this reason, after the preliminaries, we first describe a logical rule that generalizes the permanence rules of [5]. The application of our rule is subjected to a syntactical condition which subsumes the syntactical condition for the permanence rules of [5]. We use this syntactical condition to design a decision procedure that under some syntactical conditions avoids the application of non-invertible rules. The main work is to prove that the decision procedure is complete, a result that we show by exploiting the model theoretic characterization of **Int** by means of Kripke models.

2 Preliminaries

We consider the propositional language $\mathcal{L}$ based on a denumerable set of propositional variables $\mathcal{PV}$, the logical connectives $\wedge$, $\vee$, $\to$ and the logical constants $\top$ and $\bot$. We refer to [1,6,7] for details about **Int** and tableau systems. For sake of concreteness we consider the terminating calculus in Figure 1 whose rules handle *signed formulas*, namely formulas of $\mathcal{L}$ prefixed with one of the well-known signs **T** or **F**. The satisfiability of a signed formula H in a world α of a Kripke model $\underline{K}$ is defined as follows: α *realizes* H *in* $\underline{K}$ ($\underline{K}, \alpha \rhd H$) iff: (i) $H \equiv \mathbf{T}A$ and $\alpha \Vdash A$; (ii) $H \equiv \mathbf{F}A$ and $\alpha \nVdash A$. A model $\underline{K}$ realizes H ($\underline{K} \rhd H$) iff $\underline{K}, \alpha \rhd H$ for some $\alpha \in P$; a formula H is *realizable* iff $\underline{K} \rhd H$ for some Kripke model $\underline{K}$.
When possible, in the following sections we will omit notation and we write: $\alpha \rhd H$ and $\alpha \rhd \Delta$ in place of $\underline{K}, \alpha \rhd H$ and $\underline{K}, \alpha \rhd \Delta$ when it is clear from the context to which model the realizability relation refers to.

The tableau calculus of Figure 1 is adapted from the calculus presented in [9]. Our aim is to consider a simple tableau calculus to decide **Int**, whose deductions are always finite. Thus, differently from [9], calculus in Figure 1 does not handle negation and only employs the signs **T** and **F**. Calculus of Figure 1 has also similarities with Fitting's tableau calculus [6]. In the rules of the calculus we distinguish *the premise*, the set of formulas above the line and *the consequence*, the set(s) of formulas below the line that we call *conclusion(s)*, separated by a vertical line when the consequence of the rule contains two conclusions. In the premise, the *main formula of the premise* is the formula whose connectives are in evidence, the other formulas are the *minor formulas of the premise*. The formulas

in evidence in a conclusion are the *main formulas of the conclusion*. We say that a rule applies to a set of formulas Γ, if the premise of the rule can be instantiated with Γ. We implicitly always consider *duplication-free instantiations*, that is after the instantiation of the premise of a rule, the main formula premise does not occur in Δ (that is, it does not occur as minor formula premise). Given a non-atomic signed formula H, we denote with $Rule(H)$ the name of the rule whose main formula premise can be instantiated with H. A set Γ of signed formulas is *inconsistent* if $\{\mathbf{T}A, \mathbf{F}A\} \in \Gamma$ or $\mathbf{T}\bot \in \Gamma$ and Γ is *consistent* in the other cases. A tableau for a formula A is a tree obtained from the root $\{\mathbf{F}A\}$ by subsequently instantiating the premise of a rule with consistent leaves. If all the leaves of a tableau are inconsistent, then we say that the tableau is *closed*, the tableau is a *proof for* A and A *is provable*.

The rules in Figure 1 are *sound*: the realizability of the premise implies that there exists a conclusion which is realizable. It is easy to prove that calculus in Figure 1 is also *complete* for **Int**: for every formula A, A is provable iff $\mathbf{F}A$ is not realizable. To decide the provability of A it is sufficient to search for a closed tableau. An obvious algorithm tries to build a proof for A by applying the rules in all possible ways. This simple method can be improved by noticing that some rules are *invertible*: the realizability of one of the sets in the conclusion implies the realizability of the premise. Thus backtracking is not required when an invertible rule is applied. This is a well-known strategy, also employed in [6,9]. We remark that such a strategy based on the calculi in Figure 1 and in [9] is terminating. In the following sections we aim to show that further restrictions to the search space of proofs are possible. Thus it should be possible to reduce the backtrack and preserve the completeness.

Our first result is a generalization of the *permanence rules* introduced in [5] to reduce the search space. When a propositional variable p fulfills a syntactic constraint, the permanence rules allow to deduce a set where all the occurrences of p are replaced with a logical constant. The syntactic constraint is defined in terms of positive and negative occurrence of a propositional variable p in a signed formula H by the relations $p \preceq^+ H$ (p *positively occurs in* H) and $p \preceq^- H$ (p *negatively occurs in* H). Hereafter we use $\mathcal{S}$ to denote either $\mathbf{T}$ or $\mathbf{F}$. The definition of $p \preceq^l H$, with $l \in \{+, -\}$, is by induction on the structure of H: (i) $p \preceq^- \mathbf{F}p$ and $p \preceq^+ \mathbf{T}p$; (ii) $p \preceq^l \mathcal{S}\top$ and $p \preceq^l \mathcal{S}\bot$; (iii) $p \preceq^l \mathcal{S}q$, where q is any propositional variable such that $q \neq p$; (iv) $p \preceq^l \mathcal{S}(A \odot B)$ iff $p \preceq^l \mathcal{S}A$ and $p \preceq^l \mathcal{S}B$, where $\odot \in \{\wedge, \vee\}$; (v) $p \preceq^l \mathbf{F}(A \to B)$ iff $p \preceq^l \mathbf{T}A$ and $p \preceq^l \mathbf{F}B$; (v) $p \preceq^l \mathbf{T}(A \to B)$ iff $p \preceq^l \mathbf{F}A$ and $p \preceq^l \mathbf{T}B$. Given a set Δ of signed formulas, $p \preceq^l \Delta$ iff for every $H \in \Delta$, $p \preceq^l H$. In paper [5] it is proved that *permanence* rules given in the following are invertible:

$$\frac{\Delta}{\Delta[\top/p]}\ \mathbf{T}\text{-perm, provided } p \preceq^+ \Delta \qquad \frac{\Delta}{\Delta[\bot/p]}\ \mathbf{T} \to \bot\text{-perm, provided } p \preceq^- \Delta,$$

where $\Delta[A/B]$ is the result of replacing in Δ every occurrence of B with A. Intuitively, these rules state that, given a set Δ, if the syntactical constraint $p \preceq^+ \Delta$ (resp. $p \preceq^- \Delta$) is fulfilled, then it is correct to replace every occurrence of p in Δ with the logical constant $\top$ (resp. $\bot$).

3 An extension of the permanence rules

Now we aim to study a weaker condition for the correct application of rules $\mathbf{T}$-perm and $\mathbf{T} \to \bot$-perm. By exploiting such a weaker condition, in next section we introduce a decision procedure allowing to reduce the search space in proof search for **Int**. The procedure we describe does not need to employ the permanence rules as further rules of the logical apparatus. This means that the results we show are not tied neither to the calculus given in Figure 1 nor to the proof system: a decision procedure based on new tableau calculus or a sequent calculus can use our results to reduce the search space for a proof.

As first step, let us go back over the side conditions on the applicability of the permanence rules. To this aim we introduce a new notion of replacement. Let H be a signed formula and let p a propositional variable. We define *replacement in a formula H of all positive occurrences of a propositional variable p with $\top$*, the formula denoted with $H[\![\top/p]\!]$, obtained from H as follows:

- if $H = \mathbf{T}p$, then $H[\![\top/p]\!] = \mathbf{T}\top$;
- if $H = \mathbf{T}q$, with $q \in \mathcal{PV} \setminus \{p\}$, then $H[\![\top/p]\!] = H$;
- if $H = \mathbf{F}q$, then $H[\![\top/p]\!] = H$, $q \in \mathcal{PV}$;
- if $H = \mathcal{S}(A_1 \odot A_2)$, then $H[\![\top/p]\!] = \mathcal{S}(A'_1 \odot A'_2)$, with $\mathcal{S}A'_i = \mathcal{S}A_i[\![\top/p]\!]$, for $i = 1, 2$, $\mathcal{S} \in \{\mathbf{T}, \mathbf{F}\}$ and $\odot \in \{\wedge, \vee\}$;
- if $H = \mathbf{F}(A \to B)$, then $H[\![\top/p]\!] = \mathbf{F}(A' \to B')$, where $\mathbf{T}A' = \mathbf{T}A[\![\top/p]\!]$ and $\mathbf{F}B' = \mathbf{F}B[\![\top/p]\!]$;
- if $H = \mathbf{T}(A \to B)$, then $H[\![\top/p]\!] = \mathbf{T}(A' \to B')$, where $\mathbf{F}A' = \mathbf{F}A[\![\top/p]\!]$ and $\mathbf{T}B' = \mathbf{T}B[\![\top/p]\!]$.

The following are examples of replacement of q and p respectively:
$\mathbf{F}(p \to (q \vee q \to \bot))[\![\top/q]\!] = \mathbf{F}(p \to (q \vee \top \to \bot))$;
$\mathbf{F}((p \vee p \to \bot) \to q)[\![\top/p]\!] = \mathbf{F}((\top \vee p \to \bot) \to q)$.

Analogously, we define *replacement in a formula H of all negative occurrences of a propositional variable p with $\bot$*, the formula denoted with $H[\![\bot/p]\!]$ obtained from H as follows:

- if $H = \mathbf{T}q$, then $H[\![\bot/p]\!] = H$, if $q \in \mathcal{PV}$;
- if $H = \mathbf{F}p$, then $H[\![\bot/p]\!] = \mathbf{F}\bot$;
- if $H = \mathbf{F}q$, with $q \in \mathcal{PV} \setminus \{p\}$, then $H[\![\bot/p]\!] = H$;
- if $H = \mathcal{S}(A_1 \odot A_2)$, then $H[\![\bot/p]\!] = \mathcal{S}(A'_1 \odot A'_2)$, with $\mathcal{S}A'_i = \mathcal{S}A_i[\![\bot/p]\!]$, $i \in \{1, 2\}$, $\mathcal{S} \in \{\mathbf{T}, \mathbf{F}\}$ and $\odot \in \{\wedge, \vee\}$;
- if $H = \mathbf{F}(A \to B)$, then $H[\![\bot/p]\!] = \mathbf{F}(A' \to B')$, where $\mathbf{T}A' = \mathbf{T}A[\![\bot/p]\!]$ and $\mathbf{F}B' = \mathbf{F}B[\![\bot/p]\!]$;
- if $H = \mathbf{T}(A \to B)$, then $H[\![\bot/p]\!] = \mathbf{T}(A' \to B')$, where $\mathbf{F}A' = \mathbf{F}A[\![\bot/p]\!]$ and $\mathbf{T}B' = \mathbf{T}B[\![\bot/p]\!]$.

The result of $H[\![\top/p]\!]$ (resp. $H[\![\bot/p]\!]$) is a formula having the same sign of H and containing zero or more occurrences of the logical constant $\top$(resp. $\bot$). Hereafter we use $[\![]\!]$-replacement to refer to both kind of replacements defined above. We extend $[\![]\!]$-replacement to sets of signed formulas in the obvious way. If

$p \preceq^+ \Delta$ (resp. $p \preceq^- \Delta$), then $\Delta[\![\top/p]\!]$ (resp. $\Delta[\![\bot/p]\!]$) coincides with $\Delta[\top/p]$ (resp. $\Delta[\bot/p]$). Given a signed formula H and a set of signed formulas Δ, we write $H[\![]\!]$ (resp. $\Delta[\![]\!]$) to refer both formula $H[\![\top/p]\!]$ and $H[\![\bot/p]\!]$ (resp. set $\Delta[\![\top/p]\!]$ and $\Delta[\![\bot/p]\!]$). Let A be a formula, eval (A) denotes the formula obtained by applying to A the usual boolean reductions based on the meaning of the logical constants. Given a signed formula $\mathcal{S}A$, where $\mathcal{S} \in \{\mathbf{T}, \mathbf{F}\}$, the *evaluation* of a signed formula $\mathcal{S}A$, denoted as eval$(\mathcal{S}A)$, is the signed formula $\mathcal{S}$eval(A). The following is the evaluation of the formulas obtained from the previous example:
eval$(\mathbf{F}(p \to (q \vee q \to \bot))[\![\top/q]\!]) = $ eval$(\mathbf{F}(p \to (q \vee \top \to \bot))) = \mathbf{F}(p \to q)$;
eval$(\mathbf{F}((p \vee p \to \bot) \to q)[\![\top/p]\!]) = $ eval$(\mathbf{F}((\top \vee p \to \bot) \to q)) = \mathbf{F}q$.
The invertibility of the boolean simplification rules implies that $H[\![\nabla/p]\!]$ is realizable iff eval$(H[\![\nabla/p]\!])$ is realizable, with $\nabla \in \{\top, \bot\}$. The following Propositions 1 and 2 aim to give the whole picture of the relationship between the realizability of H and eval$(H[\![\nabla/p]\!])$. We start with a technical lemma:

Lemma 1. *Let H be a formula such that* eval$(H) = H$*, $H \neq \top$ and $H \neq \bot$ and let p be a propositional variable. The following holds: 1.* eval$(\mathbf{T}H[\![\nabla/p]\!]) \neq \mathbf{T}\bot$*, with $\nabla \in \{\top, \bot\}$; 2.* eval$(\mathbf{F}H[\![\nabla/p]\!]) \neq \mathbf{F}\top$*, with $\nabla \in \{\top, \bot\}$.*

The following proposition introduces the condition under which $[\![]\!]$-replacement preserves the realizability. Thus the proposition is a result about the correctness of $[\![]\!]$-replacement:

Proposition 1. *Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be a Kripke model, let $\alpha \in P$, let H be a signed formula and let p be a propositional variable: 1. if $\alpha \rhd H$ and p does not occur in* eval$(H[\![\top/p]\!])$*, then $\alpha \rhd$* eval$(H[\![\top/p]\!])$*; 2. if $\alpha \rhd H$ and p does not occur in* eval$(H[\![\bot/p]\!])$*, then $\alpha \rhd$* eval$(H[\![\bot/p]\!])$*.*

Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be a Kripke model and p a propositional variable. Following [5], we define the models $\underline{K}_p^+$ and $\underline{K}_p^-$ as follows: $\underline{K}_p^+ = \langle P, \leq, \rho, \Vdash' \rangle$, where $\Vdash' = \Vdash \cup \{(\alpha, p) \mid \alpha \in P\}$; $\underline{K}_p^- = \langle P, \leq, \rho, \Vdash' \rangle$, where $\Vdash' = \Vdash \setminus \{(\alpha, p) \mid \alpha \in P\}$. Note that, for every $\alpha \in P$, $\underline{K}_p^+, \alpha \rhd \mathbf{T}p$ and $\underline{K}_p^-, \alpha \rhd \mathbf{T}(p \to \bot)$. In following we prove that $[\![]\!]$-replacement preserves the completeness:

Proposition 2. *Let $\underline{K} = \langle P, \leq, \rho, \Vdash \rangle$ be a Kripke model, $\alpha \in P$, H a signed formula and p be a propositional variable. 1. Let us assume that p does not occur in* eval$(H[\![\top/p]\!])$*. If $\underline{K}, \alpha \rhd$* eval$(H[\![\top/p]\!])$*, then $\underline{K}_p^+, \alpha \rhd H$; 2. Let us assume that p does not occur in* eval$(H[\![\bot/p]\!])$*. If $\underline{K}, \alpha \rhd$* eval$(H[\![\bot/p]\!])$*, then $\underline{K}_p^-, \alpha \rhd H$.*

Now, let us consider the following *independence* rules:
$\dfrac{\Delta}{\Delta[\top/p]}$ $+$-indep, provided $p \notin \mathcal{PV}(\text{eval}(\Delta[\![\top/p]\!]))$ $\quad \dfrac{\Delta}{\Delta[\bot/p]}$ $-$-indep, provided $p \notin \mathcal{PV}(\text{eval}(\Delta[\![\bot/p]\!]))$
By Propositions 1 and 2 we get:

Theorem 1. *The rules* $+$-indep *and* $-$-indep *are invertible.*

The notion of $[\![]\!]$-replacement can be extended to the replacement of two or more propositional variables. Let $\underline{p} = \{p_1, \ldots, p_n\}$ and $\underline{q} = \{q_1, \ldots, q_m\}$ be two sets of propositional variables. We define

$H[\![\top/p_1,\ldots,\top/p_n,\bot/q_1,\ldots,\bot/q_m]\!] = H[\![\top/\underline{p},\bot/\underline{q}]\!]$, where $\underline{p}\cap\underline{q}=\emptyset$, as follows:

- if $H=\mathbf{T}v$, then $H[\![\top/\underline{p},\bot/\underline{q}]\!] = \begin{cases}\mathbf{T}\top \text{ if } v\in\underline{p}\\ H \quad \text{otherwise}\end{cases}$
- if $H=\mathbf{F}v$, then $H[\![\top/\underline{p},\bot/\underline{q}]\!] = \begin{cases}\mathbf{F}\bot \text{ if } v\in\underline{q}\\ H \quad \text{otherwise}\end{cases}$
- if $H=\mathcal{S}(A_1\odot A_2)$, then $H[\![\top/\underline{p},\bot/\underline{q}]\!]=\mathcal{S}(A_1'\odot A_2')$, with $\mathcal{S}A_i'=\mathcal{S}A_i[\![\top/\underline{p},\bot/\underline{q}]\!]$, for $i=1,2$, $\mathcal{S}\in\{\mathbf{T},\mathbf{F}\}$ and $\odot\in\{\wedge,\vee\}$;
- if $H=\mathbf{F}(A\to B)$, then $H[\![\top/\underline{p},\bot/\underline{q}]\!]=\mathbf{F}(A'\to B')$, where $\mathbf{T}A'=\mathbf{T}A[\![\top/\underline{p},\bot/\underline{q}]\!]$ and $\mathbf{F}B'=\mathbf{F}B[\![\top/\underline{p},\bot/\underline{q}]\!]$;
- if $H=\mathbf{T}(A\to B)$, then $H[\![\top/\underline{p},\bot/\underline{q}]\!]=\mathbf{T}(A'\to B')$, where $\mathbf{F}A'=\mathbf{F}A[\![\top/\underline{p},\bot/\underline{q}]\!]$ and $\mathbf{T}B'=\mathbf{T}B[\![\top/\underline{p},\bot/\underline{q}]\!]$.

In a similar way we can define the meaning of $H[\top/\underline{p},\bot/\underline{q}]$. The extension of these notions to sets of formulas is obvious. The rule

$$\frac{\Delta}{\Delta[\top/\underline{p},\bot/\underline{q}]}\ \pm\text{-indep, provided } (\underline{p}\cup\underline{q})\cap\mathcal{PV}(\text{eval}(\Delta[\![\top/\underline{p},\bot/\underline{q}]\!]))=\emptyset \text{ and } \underline{p}\cap\underline{q}=\emptyset$$

generalizes $+$-indep and $-$-indep and is invertible. When $(\underline{p}\cup\underline{q})\cap\mathcal{PV}(\text{eval}(\Delta[\![\top/\underline{p},\bot/\underline{q}]\!]))=\emptyset$ holds we say that the set of variables $\underline{p}\cup\underline{q}$ is independent in Δ. We remark that the intuitively obvious request $\underline{p}\cap\underline{q}=\emptyset$ is necessary to prove the invertibility of $\pm$-indep, the analogous of Proposition 2. As a matter of fact, if $v\in\underline{p}\cap\underline{q}$ we should build a Kripke model behaving as $\underline{K}_v^+$ and $\underline{K}_v^-$, which is obviously impossible.

Rule $\pm$-indep is a consequence of the definition of $[\![]\!]$-replacement and it represents a generalization of the permanence rules: under the syntactical conditions stated for rule $\pm$-indep, it is possible to replace in a set Δ the occurrences of a propositional variable p with a logical constant and such a replacement is correct also if neither $p\preceq^+\Delta$ nor $p\preceq^-\Delta$ holds. Since variables are replaced by constants, rule $\pm$-indep is a mechanism to reduce the search space. Rule $\pm$-indep can be inserted in a procedure for deciding **Int** and since permanence rules are subsumed by $\pm$-indep, they can be removed from the deductive system without any loss. To apply $\pm$-indep one has to find an independent set of propositional variables fulfilling the side condition for $\pm$-indep. This could be a computationally expansive task. We are not interested to use the rule in this way. To bound the backtracking in **Int** proof search, the decision procedure we provide checks if a given set of propositional variables is independent in a set. When the condition is fulfilled, then rule $\pm$-indep is applicable, but the important point is that in this case some backtracking steps are avoidable. Summarizing, the independence of a set of propositional variables in a set of formulas is the key point we use to avoid the backtracking.

4 Reducing backtracking in Int

Now we want to exploit $[\![]\!]$-replacement and Propositions 1 and 2 to avoid useless applications of non-invertible rules. Since we are going to avoid some rule appli-

cations, the question is to prove the decision procedure we provide is complete. To this aim we need to prove the following theorem:

Theorem 2. *Let Δ be a set of $\mathbf{T}$-signed formulas and let $\underline{K} = \langle P,\leq,\rho,\Vdash\rangle$ be a model such that $\rho \rhd \Delta$ and $\rho \Vdash p$ iff $\mathbf{T}p \in \Delta$. Let*

$$\Gamma = \Delta \cup \{\mathbf{T}p_1, \ldots, \mathbf{T}p_n, \mathbf{F}q_1, \ldots, \mathbf{F}q_m, \mathbf{T}(h_1 \to H_1), \ldots, \mathbf{T}(h_l \to H_l)\}$$

be a consistent set with $\underline{p} \cap \underline{h} = \emptyset$, where $\underline{p} = \{p_1, \ldots, p_n\}$ and $\underline{h} = \{h_1, \ldots, h_l\}$. If $(\underline{p} \cup \underline{h}) \cap \mathcal{PV}(\mathrm{eval}(\Delta[\![\top/\underline{p}, \bot/\underline{h}]\!])) = \emptyset$ holds, then: 1. Γ is realizable by the Kripke model $\underline{M} = \langle P, \leq, \rho, \Vdash_{\underline{M}}\rangle$ such that $\Vdash_{\underline{M}} = (\Vdash \cup (P \times \underline{p})) \setminus (P \times \underline{h})$; 2. if $\underline{K}, \rho \rhd \mathbf{F}(A \to B)$ and $(\underline{p} \cup \underline{h}) \cap \mathcal{PV}(\mathrm{eval}(\mathbf{F}(A \to B)[\![\top/\underline{p}, \bot/\underline{h}]\!])) = \emptyset$, then $\underline{M}, \rho \rhd \mathbf{F}(A \to B)$.

Roughly speaking, the idea to avoid backtracking as applied in following Function Bb, can be explained as follows: let S and S' be two sets of formulas and $\underline{M} = \langle P,\leq,\rho,\Vdash\rangle$ a Kripke model such that $\underline{M}, \rho \rhd S$. We are investigating under which syntactical conditions of S and S' we can change the forcing in $\underline{M}$ to get a model $\underline{M}' = \langle P, \leq, \rho, \Vdash'\rangle$ such that $\underline{M}', \rho \rhd S'$. More precisely, we are asking under which syntactical conditions the realizability of S implies the realizability of S'. This is of interest because, if an attempt to find a proof starting from S fails, then another attempt starting from another set has possibly to be done. The failed attempts witnesses that some sets are realizable by a Kripke model. Thus we wonder if the failed attempt implies that some other attempts we are going to try, for example starting from S', will fail. If we found a general criterion, computationally not expensive, to check if this case, then we can reduce the search space and speed-up the procedure: such a criterion is the side condition stated for rule $\pm$-indep. Function Bb works by locking sets of $\mathbf{T}$-signed formulas. Locked formulas are not at disposal of the backtracking steps in the sense that Function Bb does not use locked formulas as main formulas. The idea behind the locking is to mark a set a formulas having a model. This avoids the useless backtracking generated by taking the marked formulas as main formula premise of the non-invertible rule $\mathbf{T} \to\to$. To decide a formula A, the call $\mathrm{Bb}(\{\mathbf{F}A\}, \_)$ is performed, where $\_$ emphasizes that the second parameter is irrelevant. Rules are applied according to a priority which is standard: $\mathbf{T} \to\to$ and $\mathbf{F}\vee_i$ are rules of the lowest priority because if a proof is not found, they require to backtrack. Function Bb returns a proof or a structure that in completeness theorem is proved to be a Kripke model $\underline{K} = \langle P,\leq,\rho,\Vdash\rangle$ such that $\rho \rhd \Delta$. In the following, we call α-rules $\mathbf{T}\wedge$, $\mathbf{T} \to \wedge$, $\mathbf{T} \to \vee$, $\mathbf{F} \to$ and β-rules $\mathbf{T}\vee$, $\mathbf{F}\wedge$.

Function $\mathrm{Bb}(\Delta, \underline{M})$

1. If Δ is an inconsistent set, then return the proof Δ;

2. If the premise of MP can be instantiated with Δ, where $\mathbf{T}A$ and $\mathbf{T}(A \to B)$ can be locked or unlocked, then let Δ_1 be the conclusion and let $\pi = \mathrm{Bb}(\Delta_1, \underline{M})$, where $\mathbf{T}B$ is unlocked in Δ_1. If π is a proof, then return the proof $\dfrac{\Delta}{\pi}\mathrm{MP}$, otherwise return π.

3. If the premise of an α-rule can be instantiated with Δ, then let H be the main formula premise, Δ_1 the conclusion and $\pi = \mathrm{Bb}(\Delta_1, \underline{M})$. If π is a proof, then return $\dfrac{\Delta}{\pi}\,{}_{Rule(H)}$, otherwise return π.

4. If the premise of a β-rule can be instantiated with Δ, then let H be the main premise, Δ_1 and Δ_2 the conclusions and, for $i = 1, 2$ let $\pi_i = \mathrm{Bb}(\Delta_i, \underline{M})$. If π_i is a structure,

with $i \in \{1,2\}$, then return π_i, otherwise return the proof $\dfrac{\Delta}{\pi_1|\pi_2}\,Rule(H)$.

5. (i) (only backtracking on $\mathbf{F}\vee$-formula is required). If $\Delta_{\mathbf{T}}$ is not empty and all the formulas in $\Delta_{\mathbf{T}}$ are locked and $\mathbf{F}(A_1 \vee A_2) \in \Delta$, then let $\Delta_i = \Delta_{\mathbf{T}} \cup \{\mathbf{F}A_i\}$ and $\pi_i = \textsc{Bb}(\Delta_i, \underline{M})$, for $i = 1, 2$. If π_i is a proof, with $i \in \{1,2\}$, then return the proof $\dfrac{\Delta}{\pi_i}\,\mathbf{F}\vee_i$; otherwise, let $\underline{M} = \langle P_{\underline{M}}, \leq_{\underline{M}}, \rho_{\underline{M}}, \Vdash_{\underline{M}}\rangle$ and $\pi_i = \langle P_i, \leq_i, \rho_i, \Vdash_i\rangle$, for $i = 1, 2$. Return the structure $\underline{K} = \langle P, \leq, \rho, \Vdash\rangle$ defined as follows:
$P = P_{\underline{M}} \cup P_1 \cup P_2$; $\quad \leq\, = \leq_{\underline{M}} \cup \leq_1 \cup \leq_2 \cup\{(\rho,\alpha)|\alpha \in P_1 \cup P_2\}$;
$\rho = \rho_{\underline{M}}$; $\quad \Vdash\, = \Vdash_{\underline{M}} \cup \Vdash_1 \cup \Vdash_2$.

(ii) (Full standard backtracking is required) If $\Delta_{\mathbf{T}}$ contains an unlocked formula of the kind $\mathbf{T}((A \to B) \to C)$ or, all formulas in $\Delta_{\mathbf{T}}$ are unlocked and $\mathbf{F}(A \vee B) \in \Delta$, then return $\textsc{BackT}(\Delta, \underline{M})$.

(iii) If $\Delta_{\mathbf{T}}$ contains both locked and unlocked formulas (note that the unlocked formulas are atomic or of the kind $\mathbf{T}(p \to B)$), then let $\Phi = \{\mathbf{F}(A \to B)|\mathbf{T}((A \to B) \to C) \in \Delta\}$, let $\Delta_U = \{\mathbf{T}A \in \Delta|\mathbf{T}A \text{ is unlocked }\}$, $\underline{p} = \{p|\mathbf{T}p \in \Delta_U\}$, $\underline{q} = \{q|\mathbf{T}(q \to H) \in \Delta_U\}$ and let $\underline{K} = \langle P, \leq, \rho, \Vdash\rangle$ be the structure defined as follows:

$P = P_{\underline{M}}$; $\rho = \rho_{\underline{M}}$; $\leq\, = \leq_{\underline{M}}$; $\Vdash\, = (\Vdash_{\underline{M}} \setminus (P \times \underline{q})) \cup (P \times \underline{p})$.

(A) If $\mathbf{F}(A_1 \vee A_2) \notin \Delta$ and $(\underline{p} \cup \underline{q}) \notin \mathcal{PV}(\text{eval}(\Delta[\![\top/\underline{p}, \bot/\underline{q}]\!]))$, then return $\underline{K}$.

(B) If $\mathbf{F}(A_1 \vee A_2) \in \Delta$ and $(\underline{p} \cup \underline{q}) \notin \mathcal{PV}(\text{eval}((\Delta \cup \Phi)[\![\top/\underline{p}, \bot/\underline{q}]\!]))$, then for $i = 1, 2$, let $\pi_i = \textsc{Bb}(\Delta_{\mathbf{T}} \cup \{\mathbf{F}A_i\}, \underline{K})$, where in the recursive call all formulas in $\Delta_{\mathbf{T}}$ are locked. If there exists $i \in \{1,2\}$ such that π_i is a proof, then return the proof $\dfrac{\Delta}{\pi_i}\,\mathbf{F}\vee_i$, else let $\pi_i = \langle P_i, \leq_i, \rho_i, \Vdash_i\rangle$; return the structure $\langle P', \leq', \rho, \Vdash'\rangle$ defined as follows:
$P' = P \cup P_1 \cup P_2$; $\leq' = \leq \cup \leq_1 \cup \leq_2 \cup\{(\rho,\alpha)|\alpha \in P_i\}$; $\Vdash' = \Vdash \cup \Vdash_1 \cup \Vdash_2$.

(C) Unlock the formulas in Δ and return $\textsc{Bb}(\Delta, \_)$.

6. (if we are here $\mathbf{F}(A \vee B) \notin \Delta$). If all the formulas in $\Delta_{\mathbf{T}}$ are locked, then (i) return $\underline{M}$, else (ii) return $\langle \{\rho\}, \{(\rho,\rho)\}, \rho, \{(\rho,p) \mid \mathbf{T}p \in \Delta\}\rangle$.

End Function Bb.

Function $\textsc{BackT}(\Delta, \underline{M})$
Let $\{\mathbf{T}((A_i \to B_i) \to C_i)\}_{1 \leq i \leq n} = \{\mathbf{T}((A \to B) \to C) \in \Delta\}$;
for $i = 1, \ldots, n$
- let $\phi_i = \textsc{Bb}((\Delta \setminus \{\mathbf{T}((A_i \to B_i) \to C_i)\}) \cup \{\mathbf{T}C_i\}, \underline{M})$, where in the recursive call $\mathbf{T}C_i$ is not locked and the locking of the other formulas is left unchanged;
- if ϕ_i is a structure, then return ϕ_i.

Unlock all the formulas in Δ;
for $i = 1, \ldots, n$
- let $\pi_i = \textsc{Bb}((\Delta_{\mathbf{T}} \setminus \{\mathbf{T}((A_i \to B_i) \to C_i)\}) \cup \{\mathbf{T}A_i, \mathbf{F}B_i, \mathbf{T}(B_i \to C_i)\}, \_)$;
- if π_i is a proof, then return the proof $\dfrac{\Delta}{\pi_i|\phi_i}\,\mathbf{T}\to\to$.

Given the structures $\pi_i = \langle P_i, \leq_i, \rho_i, \Vdash_i\rangle$, with $i = 1, \ldots, n$, define $\underline{K} = \langle P, \leq, \rho, \Vdash\rangle$ as follows:
$P = \rho \cup_{i=1}^{n} P_i$; $\leq\, = \cup_{i=1}^{n} \leq_i \cup\{(\rho,\alpha)|\alpha \in P\}$; $\Vdash\, = \cup_{i=1}^{n} \Vdash_i \cup\{(\rho,p)|\mathbf{T}p \in \Delta\}$.
If $\mathbf{F}(A_1 \vee A_2) \notin \Delta$, then return $\underline{K}$.

For $i = 1, 2$
- let $\pi_{n+i} = \textsc{Bb}(\Delta_{\mathbf{T}} \cup \{\mathbf{F}A_i\}, \underline{K})$, where in the recursive call all the formulas in $\Delta_{\mathbf{T}}$ are locked; if π_{n+i} is a proof, then return the proof $\dfrac{\Delta}{\pi_{n+i}}\,\mathbf{F}\vee_i$.

Return the structure $\langle P, \leq, \rho, \Vdash \rangle$ defined as follows:
$P = \rho \cup_{i=1}^{n+2} P_i$; $\leq = \cup_{i=1}^{n} \leq_i \cup \{(\rho, \alpha) | \alpha \in P\}$; $\Vdash = \cup_{i=1}^{n+2} \Vdash_i \cup \{(\rho, p) | \mathbf{T}p \in \Delta\}$.
End Function BacktT.

Function Bb locks the $\mathbf{T}$-formulas after the application of rule $\mathbf{F}\vee_i$. When this stage is reached, the realizability of $\Delta_{\mathbf{T}}$ has been proved by backtracking steps involving the non-invertible conclusion of the rule $\mathbf{T} \to\to$ and $\underline{M}$ is the model realizing $\Delta_{\mathbf{T}}$. The formulas in $\Delta_{\mathbf{T}}$ will be unlocked when a new $\mathbf{T} \to\to$-formula is introduced. If a set Γ only containing locked formulas, signed atomic formulas and $\mathbf{T}(p \to H)$-formulas, with $\mathbf{T}p \notin \Gamma$, is gotten, then, by Theorem 2, we can decide if the realizability of $\Delta_{\mathbf{T}}$ implies the realizability of Γ. This check is performed in Step 5(iii)(A). By construction $\Delta_{\mathbf{T}} \subseteq \Gamma$ and $\Delta_{\mathbf{T}}$ coincides with the formulas of Γ that are locked. Function Bb contains a call to Function BacktT. Function BacktT implements the backtracking mechanism necessary to handle formulas of the kind $\mathbf{T}((A \to B) \to C)$ and $\mathbf{F}(A \vee B)$. This is a standard phase, necessary to guarantee the completeness of the decision procedure. If after all possible instantiations of the premise of $\mathbf{T} \to\to$ with Δ no proof is found, then the structure $\underline{K}$ is a Kripke model whose root realizes $\Delta_{\mathbf{T}} \cup \{\mathbf{F}(A \to B) | \mathbf{T}((A \to B) \to C) \in \Delta\}$. At this point, Bb instantiates, if possible, the premise of $\mathbf{F}\vee_1$ and $\mathbf{F}\vee_2$ with Δ. In the subsequent applications the formulas in $\Delta_{\mathbf{T}}$ are locked, thus $\mathbf{T} \to\to$-formulas are not used as main formulas. This is also the strategy employed by sequent calculus LJQ [2]. The strategy of Bb diverges from LJQ by the fact that after an application of $\mathbf{F} \to$, LJQ unlocks the formulas, whereas Bb keeps them locked until the only applicable rules are $\mathbf{T} \to\to$ or $\mathbf{F}\vee$. This corresponds to reach Step 5(ii) or 5(iii). If a new $\mathbf{T} \to\to$ formula appears, then Step 5(ii) is performed and BacktT is called. In BacktT, when the invertible conclusion of rule $\mathbf{T} \to\to$ is handled, the locking of the minor premises is left unchanged. When the non-invertible conclusion of $\mathbf{T} \to\to$ is used the locked formulas of Δ become unlocked, thus Bb behaves as LJQ. If Step 5(iii) is reached, then it means that subsequently to the locking of the formulas, new $\mathbf{T}$-formulas have been added, but they are atomic or of the kind $\mathbf{T}(p \to A)$ only. In Steps 5(iii)(A) and 5(iii)(B) function Bb attempts to avoid backtracking steps. Bb performs a purely syntactic check involving $[\![]\!]$-replacement. In practice, Bb checks if the side condition on the applicability of rule $\pm$-indep is fulfilled. It has to be noted that the check in Step 5(iii)(A) is different from the check in Step 5(iii)(B). To prove the completeness for the case of Step 5(iii)(B) we needed to know that the model $\underline{K}$ obtained from $\underline{M}$ fulfills the property that for every $\mathbf{T}((A \to B) \to C) \in \Delta$, the root of $\underline{K}$ realizes $\mathbf{F}(A \to B)$. Since by hypothesis the given model $\underline{M}$ fulfills such a property, the check in Step 5(iii)(B) aims to prove that for every $\mathbf{T}((A \to B) \to C) \in \Delta$, the realizability $\mathbf{F}(A \to B)$ is independent of the forcing of the propositional variables $\{p_1, \ldots, p_n, q_1, \ldots, q_m\}$. Finally we notice that the check in Step 5(iii)(A) and Step 5(iii)(B) is only based on the locking and the formulas in the set Δ at hand.

Let Δ be a set of formulas. In the next theorem we show that (1) if Δ does not contain locked formulas and $\text{Bb}(\Delta, \_)$ returns a structure $\underline{K}$, then $\underline{K}$ is a Kripke model such $\underline{K} \rhd \Delta$ and Bb does not require any further information;

(2) if Δ contains some locked formula, $\underline{M}$ is a Kripke model that realizes $\Delta_{\mathbf{T}}$ and $\textsc{Bb}(\Delta, \underline{M})$ returns a structure $\underline{K}$, then $\underline{K} \rhd \Delta$ and $\underline{K}$ is possibly built on the Kripke model $\underline{M}$:

Theorem 3 (Completeness). *Let Δ be a set of formulas. 1. If $\Delta_{\mathbf{T}}$ does not contain locked formulas and $\textsc{Bb}(\Delta, \underline{M})$ returns a structure $\underline{K}$, then $\underline{K} \rhd \Delta$ and $\underline{K}$ is defined independently of $\underline{M}$; 2. if $\Delta_{\mathbf{T}}$ contains a subset Γ of locked formulas and $\underline{M} = \langle P_{\underline{M}}, \leq_{\underline{M}}, \rho_{\underline{M}}, \Vdash_{\underline{M}} \rangle$ is a Kripke model such that $\underline{M} \rhd \Gamma$, for every $\mathbf{T}((A \to B) \to C) \in \Gamma$, there exists $\alpha \in P_{\underline{M}}$ such that $\rho_{\underline{M}} \neq \alpha$ and $\alpha \rhd \mathbf{T}A, \mathbf{F}B$ and $\rho_{\underline{M}} \Vdash p$ iff $\mathbf{T}p \in \Gamma$, then $\textsc{Bb}(\Delta, \underline{M})$ returns a structure $\underline{K}$ such that $\underline{K} \rhd \Delta$.*

5 Conclusions and Future Works

We have presented a new optimization rule called $\pm$-indep that generalizes the permanence rules given in [5]. By using a general result on the correctness for $\pm$-indep, we have introduced a criterion to bound the non-determinism in **Int** proof search. Our results do not depend on a particular tableau calculus or proof system and do not modify the logical apparatus a decision procedure is based on. As an example, our results can also be used by decision procedures based on proof systems obeying the subformula property.

We are working on more advanced decision procedures based on the results presented here, where the backtracking is bounded in more cases, also using information external to the set at hand and returned by the recursive calls. Moreover, apart the extension to other propositional intermediate logics, we are interested to investigate the application of these ideas to logics that require backtracking to be decided, such as some modal and temporal logics.

References

1. A. Chagrov and M. Zakharyaschev. *Modal Logic*. Oxford University Press, 1997.
2. R. Dyckhoff and S. Lengrand. LJQ: A strongly focused calculus for Intuitionistic logic. In A. Beckmann, U. Berger, B. Löwe, and J. V. Tucker, editors, *CiE*, vol. 3988 of *LNCS*, pages 173–185. Springer, 2006.
3. M. Ferrari, C. Fiorentini, and G. Fiorino. A tableau calculus for propositional Intuitionistic logic with a refined treatment of nested implications. *JANCL*, 19(2):149–166, 2009.
4. M. Ferrari, C. Fiorentini, and G. Fiorino. FCube: An efficient prover for Intuitionistic propositional logic. In C. G. Fermüller and A. Voronkov, editors, *LPAR*, vol. 6397 of *LNCS*, pages 294–301. Springer, 2010.
5. M. Ferrari, C. Fiorentini, and G. Fiorino. Simplification rules for Intuitionistic propositional tableaux. *TOCL*, 13(2):14, 2012.
6. M.C. Fitting. *Intuitionistic Logic, Model Theory and Forcing*. North-Holland, 1969.
7. R. Hähnle. Tableaux and related methods. In J.A.Robinson and A.Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001.
8. J. Hudelmaier. An $O(n \log n)$-SPACE decision procedure for Intuitionistic propositional logic. *JLC*, 3(1):63–75, 1993.
9. P. Miglioli, U. Moscato, and M. Ornaghi. Avoiding duplications in tableau systems for Intuitionistic logic and Kuroda logic. *IGPL*, 5(1):145–167, 1997.

Graphs realised by r.e. equivalence relations*

Alexander Gavruskin[1], Sanjay Jain[2],
Bakhadyr Khoussainov[1] and Frank Stephan[3]

[1] Department of Computer Science, The University of Auckland, New Zealand
a.gavruskin@auckland.ac.nz and bmk@cs.auckland.ac.nz
[2] School of Computing, National University of Singapore
Singapore 117417, Republic of Singapore
sanjay@comp.nus.edu.sg
[3] Department of Mathematics, National University of Singapore
10 Lower Kent Ridge Road, Singapore 119076, Republic of Singapore
fstephan@comp.nus.edu.sg

Abstract. We investigate dependence of recursively enumerable graphs on the equality relation which is fixed to a specific r.e. equivalence relation on ω. In particular we compare r.e. equivalence relations in terms of graphs they permit to represent. This defines a partial order that depends on classes of graphs under consideration. We show that for various classes of graphs, there are minimal and maximal elements in the corresponding ordering of r.e. equivalence relations.

1 Introduction

Recursively enumerable (r.e.) structures are given by a domain, recursive functions representing basic operators in the structure, and some recursively enumerable predicates, among which there is a predicate E representing the equality relation in the structure. When E is fixed, various algebraic properties of r.e. structures with the equality relation E depend heavily on the equivalence relation E. Furthermore, various computability-theoretic properties of E depend on algebraic properties of structures in which the equality relation is E. For example, Novikov constructed a finitely generated group with undecidable word-problem; in other words, there is a group which can be represented using an r.e. nonrecursive equivalence relation E (as equality of the group) but not using a recursive equivalence relation E. On the other hand, for Noetherian rings [18], Baur [2] showed that every r.e. Noetherian ring is a recursive ring, implying that the

* A. Gavruskin is supported by the FRDF grant of the University of Auckland and by the Federal Target Program "Research and Training Specialists in Innovative Russia, 2009–2013", Contract No 16.740.11.0567; S. Jain is supported in part by NUS grants C252-000-087-001 and R252-000-420-112; B. Khoussainov is partially supported by Marsden Fund of the Royal Society of New Zealand and he did part of this work while on Sabbatical Leave to the School of Computing, National University of Singapore; F. Stephan is supported in part by NUS grant R252-000-420-112.

underlying equality E is always a recursive relation. So only recursive equality relations E can be used to represent Noetherian rings.

Our aim is to investigate recursively enumerable graphs emphasising the role of the equivalence relation E representing the equality. In the paper [14] we initiated this program and studied general properties of r.e. structures, particularly various classes of algebras and linear orders. In this paper, we study recursively enumerable graphs, their properties, and their dependence on the equality relation. Later we will define various classes of graphs, but for the meantime for the reader by graph we mean a set of vertices together with a set of edges between the vertices where self-loops are allowed.

In this paper, we *always* assume that our equivalence relations E are r.e. equivalence relations on the set of natural numbers ω. We note that Ershov [7], and following him Odifreddi [19], call r.e. equivalence relations positive equivalence relations.

Let E be an r.e. equivalence relation on ω. We say that an n-ary relation R on ω *respects* E if for all $x_1, y_1, x_2, y_2, \ldots, x_n, y_n \in \omega$ such that $(x_1, y_1), \ldots, (x_n, y_n) \in E$ we have $R(x_1, \ldots, x_n)$ if and only if $R(y_1, \ldots, y_n)$. Note that if $n = 1$, then R is simply a unary relation of ω, and R respects E if and only if R is a union of E-equivalence classes.

Since we consider graphs, our relations R will be binary and we denote them by $Edge$. Thus, an r.e. binary relation $Edge \subseteq \omega^2$ *respects* E if for all $x_1, y_1, x_2, y_2 \in \omega$ such that $(x_1, y_1) \in E$ and $(x_2, y_2) \in E$ we have $Edge(x_1, x_2)$ if and only if $Edge(y_1, y_2)$. If $Edge \subseteq \omega^2$ respects E then $Edge$ induces a binary relation on the quotient ω/E. We abuse notation in this paper and denote the induced binary relation on ω/E by $Edge$ itself.

Definition 1. Let E be an r.e. equivalence relation.

1. An *E-graph* is a structure of the form $(\omega/E; Edge)$, where $Edge$ is a symmetric, irreflexive and r.e. binary relation respecting E.
2. An *E-pseudograph* is a structure of the form $(\omega/E; Edge)$, where $Edge$ is a symmetric and r.e. binary relation respecting E.

We say that a graph (pseudograph) is *recursively enumerable* if it is an E-graph (pseudograph) for some r.e. equivalence relation E.

For graphs and pseudographs, we sometimes use $\{u, v\}$ to represent the edge (u, v) in the graph (since there is no sense of direction on the edges). For an equivalence relation E, we write $[x]_E$ or $E(x)$ to denote the equivalence class of x. Sometimes we omit the index E. Similarly, if $Edge$ is the edge relation in a graph $\mathcal{G} = (V, Edge)$, then by $Edge(v)$ we denote the set $\{v' \in V \mid Edge(v, v')\}$. For E-graphs $(\omega/E; Edge)$, when there is no confusion, we interchange our notation and might write $Edge(x, y)$ instead of $Edge([x]_E, [y]_E)$ or vice versa.

Let C be a class of graphs (pseudographs), where we identify graphs up to isomorphism. Given an r.e. equivalence relation E we would like to single out those graphs in the class C that are isomorphic to E-graphs, as given in the following definition.

Definition 2. Given an r.e. equivalence relation E and a graph (pseudograph) $\mathcal{G}$, we say that E *realises* $\mathcal{G}$ iff there is an r.e. relation $Edge$ such that $Edge$ respects E and $\mathcal{G}$ is isomorphic to $(\omega/E; Edge)$. If E does not realise $\mathcal{G}$, then we say that E *omits* $\mathcal{G}$. We let $\mathcal{K}_C(E)$ denote all those graphs from C which are realised by E.

1.1 Examples

Example 3. If ω/E is finite, then a graph $\mathcal{G} = (V; Edge)$ belongs to $\mathcal{K}_{Graph}(E)$ if and only if the cardinality of V equals the cardinality of ω/E.

From now on, *all our graphs will be infinite graphs*, that is, graphs whose set of vertices is an infinite set. Thus, we often assume (without explicitly stating it) that all E considered in this paper are infinite (that is, ω/E is infinite).

Example 4. Let E be the identity relation id_ω on ω. Then the class $\mathcal{K}_{Graph}(E)$ consists of all graphs $(\omega; Edge)$ where $Edge$ is an r.e. set of unordered pairs. In particular, this class contains all recursive graphs.

Example 5. Let $X \subseteq \omega$ be a r.e. set. Consider the following relation $E(X)$:

$$E(X) = \{(x, y) \mid x = y\} \cup \{(x, y) \mid x, y \in X\}.$$

Each equivalence class of $E(X)$ is either a singleton $\{i\}$ where $i \notin X$ or is the set X itself.

Example 6. A *complete graph* is a graph that has edges between all pairs x, y of its vertices, where $x \neq y$. We call a pseudograph $\mathcal{G}$ an *n-complete pseudograph* if $\mathcal{G}$ has exactly n vertices with self-loops and has edges between all pairs x, y of its vertices, where $x \neq y$. Call $\mathcal{G}$ a *fully complete pseudograph* if there exists an edge between any pair of elements of $\mathcal{G}$. We observe the following:

1. Every r.e. equivalence relation realises a fully complete pseudograph;
2. For each $n \in \omega$ there exists an equivalence relation E such that E realises a k-complete pseudograph if and only if $n \leqslant k$.

Example 7. If an infinite graph $\mathcal{G}$ has finitely many edges then every r.e. equivalence relation E realises $\mathcal{G}$.

The example above shows that for every E the class $\mathcal{K}_{Graph}(E)$ is not empty.

Definition 8. The *transversal* of a recursively enumerable equivalence relation E, denoted by $tr(E)$, is the set $\{n \mid \forall x\, [x < n \to (x, n) \notin E]\}$.

Thus, the transversal $tr(E)$ is the set of all minimal elements taken from the equivalence classes of E. It is easy to see that E is Turing equivalent to $tr(E)$.

Recall that a set X of natural numbers is *hyperimmune* if there does not exist a recursive function g such that $g(i) \geqslant x_i$ for all i, where $x_0 < x_1 < x_2 < \ldots$ and $X = \{x_0, x_1, \ldots\}$. We also say that a set X is *hypersimple* if X is

recursively enumerable and its complement is hyperimmune [20]. We call a graph $\mathcal{G} = (V, Edge)$ *locally finite* if the set $Edge(v)$ is finite for all $v \in V$. A locally finite E-graph $\mathcal{G} = (\omega/E, Edge)$ is *strongly locally finite* if there exists a recursive function, which we call a *witness function*, that given an $n \in \omega$ produces a tuple $m_1, \ldots, m_k$ such that the following properties hold:

1. $Edge([n], [m_i])$ is true for $i = 1, \ldots, k$,
2. For every y such that $Edge([n], [y])$ there exists an m_i for which $(y, m_i) \in E$.

A graph $\mathcal{G} = (V, Edge)$ is called *absolutely locally finite* iff every connected set of vertices is finite.

1.2 Reducibilities

The definition of the class $\mathcal{K}_C(E)$ depends on two parameters: the class C of graphs (pseudographs) and the equivalence relation E. When we fix an r.e. equivalence relation E, the class $\mathcal{K}_C(E)$ calls for a description of those graphs from C that can be realised over E. From this point of view the class $\mathcal{K}_C(E)$ represents a graph-theoretic content of the universe ω/E. When we fix a class C of graphs, one considers those equivalence relations E that realise graphs from C. The collection of these equivalence relations can be viewed as computability-theoretic content of the class C. These observations call for the investigation of the relationship between r.e. equivalence relations in terms of graphs (from the class C) they realise. Formally, this is explained through the following definitions (also, see [14]).

Definition 9. Let C be a class of graphs (pseudographs) and let E_1 and E_2 be r.e. equivalence relations. We say E_1 is *C-reducible to* E_2, written $E_1 \leqslant_C E_2$, iff every graph in C realised by E_1 is also realised by E_2. In particular, we have the following reductions when C is the classes of pseudographs and graphs.

1. $E_1 \leqslant_{Pgraph} E_2$ iff all pseudographs realised by E_1 are realised by E_2;
2. $E_1 \leqslant_{Graph} E_2$ iff all graphs realised by E_1 are realised by E_2.

Sometimes we use a terminology borrowed from recursion theory. For instance, similar to m-degrees or Turing degrees, we say that E_1 and E_2 have the same *C-degree*, written $E_1 \equiv_C E_2$, iff $E_1 \leqslant_C E_2 \wedge E_2 \leqslant_C E_1$. The reducibility $\leqslant_C$ naturally induces the partial order on the set of all C-degrees. Without much confusion we use the same symbol $\leqslant_C$ to denote this partial order on C-degrees. Thus, there are two lines of investigation. One is to study basic properties of the partial order $\leqslant_C$ on the set of all C-degrees. The other is to investigate the graphs from $\mathcal{K}_C(E)$ by selecting various classes C of graphs. In this paper we initiate the study in both directions.

Most reducibilities (if not all) on equivalence relations and sets that have been studied aim to capture recursion-theoretic and set-theoretic complexities between equivalence relations. Typically a reduction from E to E' tells us that E' is a harder problem to solve than E. For instance, Turing reducibility from

E to E' implies that by having an oracle with access to E' we can design an algorithm that decides E. In contrast, our reducibilities given in Definition 9 aim to compare equivalence relations E and E' in terms of their algebraic content, – namely the classes $\mathcal{K}_C(E)$ and $\mathcal{K}_C(E')$ that they represent.

1.3 Connections to related work

The paper [14] initiated the study of E-structures in general setting. In particular, it investigated an important class $\mathcal{L}$ of pseudographs, namely the class of linearly ordered sets (where $(x, y) \in Edge \Leftrightarrow x \leqslant y$). It obtained some basic results about the partial order $\leqslant_{\mathcal{L}}$ and built certain equivalence relations E for which the classes $\mathcal{K}_{\mathcal{L}}(E)$ can easily be described. For instance, it constructed equivalence relations E that realise only n many linear orders, where n is fixed [14]. It also characterised some classes of linear orders that can be realised by equivalence relations of type $E(X)$. The paper [14] also studied the C-reducibility in the case when C consists of universal algebras or some of its sub-classes. Other examples of work on r.e. equivalence relations are the recent papers of Fokina, Friedman and Törnquist [12]. We also mention Bernardi and Sorbi [3,4] as well as Ershov [7,8] who studied various reducibilities between equivalence relations. Some of the reducibilities have been revisited due to the work of Fokina and Friedman [9,10,11,12] and, later, one such reducibility became known as FF-reducibility. This reducibility is roughly a many-one reducibility between r.e. equivalence relations. Below we mention the definition of FF-reducibility in order to relate it to this paper. Friedman and Fokina defined their reducibility in a more general context; however, in the context of this paper we need a special case when the two equivalence relations compared have the domain ω.

Now we recall FF-reducibility. For recursively enumerable equivalence relations E_1, E_2 on ω, we say that E_1 is *FF-reducible to* E_2, written $E_1 \leqslant_{FF} E_2$, iff there exists a recursive function f such that for all $x, y \in \omega$ we have $[x\, E_1\, y \Leftrightarrow f(x)\, E_2\, f(y)]$. This naturally induces the equivalence relation $\equiv_{FF}$ between E_1 and E_2 given as $E_1 \leqslant_{FF} E_2 \wedge E_2 \leqslant_{FF} E_1$. In this case it is said that E_1 and E_2 have the same *FF-degree*. It is not hard to see that $\leqslant_{FF}$ has the largest element among all the FF-degrees [1,12,13].

One can also consider the following equivalence relation $\sim_{FF}$ between equivalence classes. We say that E_1 and E_2 are *$\sim_{FF}$-equivalent*, written $E_1 \sim_{FF} E_2$, iff there exists a recursive function f witnessing $E_1 \leqslant_{FF} E_2$ such that all equivalence classes of E_2 appear in the range of f. Note that $\sim_{FF}$ is an equivalence relation that implies $\leqslant_{FF}$. When comparing $\sim_{FF}$ with $\equiv_{FF}$, it turns out that $\sim_{FF}$ is a more restrictive condition than $\equiv_{FF}$, namely $\equiv_{FF}$ does not always imply $\sim_{FF}$. This stands in contrast to one-one reducibility between r.e. subsets of ω [20]. We also note that if X_1, X_2 are two infinite r.e. sets then for the equivalence relations $E(X_1)$ and $E(X_2)$ (defined in Example 5) we have the following: $E(X_1) \leqslant_{FF} E(X_2)$ iff $X_1 \leqslant_1 X_2$ [6,8,17]. Hence, FF-reducibility is nearer to one-one reducibility than to many-one reducibility between r.e. subsets of ω. Coskey, Hamkins and Miller [5,6] and Gao and Gerdes [13] also contributed to the study of r.e. equivalence relations and FF-reducibility. Finally, we mention

a recent paper by Andrews, Lempp, Miller, Ng, San Mauro and Sorbi [1] that presents a comprehensive study of FF-reducibility between r.e. equivalence relations, in particular, answering several questions posed in the work of Gao and Gerdes [13].

2 Isles

Let C be a class of pseudographs (graphs). Intuitively, there is not much connection between $\leqslant_C$-reducibility and FF-reducibility on r.e. equivalence relations. In fact, we observe that the definitions of $\leqslant_{FF}$ and $\leqslant_C$ imply that FF-reducibility is an arithmetic definition while $\leqslant_C$-reducibility is a Σ^1_1-definition. However, in this section we show that for some natural classes of pseudographs, one might find connections between these two reducibilities. In this section we introduce such a class. We call pseudographs from this class isles. Here is the definition of isles.

Definition 10. An *isle* or an *island graph* is a pseudograph which has infinitely many isolated vertices. Formally, an isle is a pseudograph $(V, Edge)$ with a countable vertex set V such that there are infinitely many vertices $x \in V$ satisfying $(x, y) \notin Edge$ for all $y \in V$. Denote the class of all isles by $Isle$.

Now we can recast Definition 9 for the class $Isle$ of all isles. Namely, we say that $E \leqslant_{Isle} E'$ iff every isle realised by E is also realised by E'. As we mentioned above, the importance of this class of pseudographs stems from the fact that $Isle$-reducibility can, in some ways, be related to the FF-reducibility. Furthermore, we give a characterisation of all the isles that can be realised by all infinite r.e. equivalence relations; the characterisation involves a graph-theoretic concept of clique graphs. In addition, we construct natural examples of r.e. equivalence relations which only realise these isles. We will also prove that there are the least and the greatest $Isle$-degrees.

In recursion theory, often for the partially ordered set $(P; \leqslant)$ given by some reducibility $\leqslant$, the *greatest* element is called *universal*. Intuitively, universal degrees represent the hardest problems to which other problems can be reduced. For instance, the FF-reducibility has a universal degree.

Theorem 11. *If $E \leqslant_{FF} E'$ then $E \leqslant_{Isle} E'$. Moreover, E' is $Isle$-universal if and only if E' is FF-universal.*

We now show that $Isle$-reducibility has the least element. We also construct an example of an infinite chain in the set of $Isle$-degrees. We start with the following definition that singles out those isles for which the set of edges is finite.

Definition 12. We call an isle *finitary* iff there are only finitely many vertices which are connected to other vertices or themselves. That is, a graph $(V, Edge)$ is a *finitary isle* iff the set $\{x \in V \mid \exists y \in V\, [(x, y) \in Edge]\}$ is finite and the set V is infinite. If an isle is not finitary then we call it an *infinitary isle*.

Note that every recursively enumerable equivalence relation E realises all finitary isles. The next result shows that the least element with respect to $Isle$-reducibility is determined by the class of all finitary isles. To prove this, we recall cohesive sets. Namely, an infinite set Z is *cohesive* if no r.e. set X exists that splits Z into two infinite subsets, that is, no r.e. X exists such that both $Z \cap X$ and $Z \cap (\omega - X)$ are infinite. *Maximal* sets are recursively enumerable sets whose complements are cohesive. Maximal sets are a well-known topic studied in recursion theory [19,20].

Theorem 13. *If the transversal $tr(E)$ of an equivalence relation E is a cohesive set then the equivalence relation E realises only finitary isles. In particular, for every maximal set X, the equivalence relation $E(X)$ is the least element with respect to $Isle$-reducibility.*

Note that the $Isle$-least element contains infinitely many FF-degrees. The reason is that if X and Y are maximal sets of different Turing degrees then $E(X)$ and $E(Y)$ are FF-incomparable yet $E(X)$ and $E(Y)$ are $Isle$-equivalent. Hence, recursion-theoretically, two equivalence relations realising the same isles might be quite different.

Proposition 14. *The set of all $Isle$-degrees contains an infinite chain.*

We now characterise equivalence relations that realise infinitary isles in terms of clique which is adapted to the concept of isles.

Definition 15. A *clique-isle* is an isle for which there is a set C such that (x, y) is an edge of the isle iff $x, y \in C$ and $x \neq y$. A *full-clique-isle* is an isle for which there is a set C such that (x, y) is an edge of the isle iff $x, y \in C$.

Note that both clique-isles and full-clique-isles (over a graph with countably many vertices) are uniquely determined by the cardinality of C and their first order theories are $\aleph_0$-categorical. It turns out, as we prove in the theorem below, those equivalence relations that realise infinitary clique isles can be characterised in terms of FF-reducibility.

Theorem 16. *An equivalence relation E realises an infinitary clique-isle iff $id_\omega \leqslant_{FF} E$.*

Proposition 17. *There exists an r.e. equivalence relation E that realises an infinitary full-clique isle but omits an infinitary clique isle.*

Theorem 18. *For every r.e. equivalence relation E, the following statements are equivalent:*

(a) *E realises an infinitary isle;*
(b) *E realises the infinitary full-clique-isle;*
(c) *There is an r.e. set X respecting E such that X contains infinitely many E-equivalence classes and leaves out infinitely many E-equivalence classes.*

In Theorem 13 we proved that the partial order of the $Isle$-degrees has the least element. Our next theorem shows that there is an atom in this partial order. Recall that an *atom* in a partially ordered set $(X, \leqslant)$ with the least element z is a member a of X such that $a \neq z$ and no x exists strictly between z and a. Note that the theorem proves even more. Namely, not only the $Isle$-degrees have an atom, but also this atom is a lower bound of all non-zero $Isle$-degrees, which implies that the atom is unique. This in particular shows that the $Isle$-degrees are not dense.

Theorem 19. *There exists an r.e. equivalence relation E such that E realises an infinitary isle and for every r.e. equivalence relation E', if E' realises an infinitary isle then $E \leqslant_{isle} E'$.*

3 Partition graphs

In this section we consider another class of graphs that we call partition graphs. We denote this class by $Part$. As for isles we provide several characterisation results and examples. An importance of this class of graphs is that the $Part$-reducibility induced by $Part$ behaves somewhat orthogonally to the $Isle$-reducibility induced by isles.

Definition 20. A graph $\mathcal{G} = (V, Edge)$ is called a *partition graph* if and only if there is a (finite or infinite) partition $A_0, A_1, \ldots$ of the set of vertices such that $\{x, y\} \in Edge$ iff there is no k for which $x, y \in A_k$.

In the definition above, for the partition graph $\mathcal{G}$, the sets $A_0, A_1, \ldots$ stated are called the anti-clique components of $\mathcal{G}$. We denote the class of partition graphs with $Part$; this induces the $Part$-reducibility on r.e. equivalence relations.

The graph $(V, Edge)$ in which all nodes are completely isolated, that is $Edge = \varnothing$, is clearly a partition graph. The anti-clique component of this graph is V itself. An infinite complete graph is also an example of a partition graph. The anti-clique components of the complete graph are singletons. These are two examples of trivial partition graphs. However, these two trivial partition graphs are complete opposites of each other in terms of equivalence relations that realise them.

Theorem 21. *The following statements are true:*

1. *Every equivalence relation E realises the trivial partition graph in which all vertices are completely isolated;*
2. *An equivalence relation E realises an infinite complete graph if and only if E is recursive.*

In the study of r.e. equivalence relations, Maltsev [16] introduced the concept of precomplete equivalence relation and studied their properties. An r.e. equivalence relation E is *precomplete* iff for every partial-recursive function $\psi : \omega \to \omega$ there is a total-recursive function f such that for all $n \in dom(\psi)$, we have $\psi(n)\, E\, f(n)$.

Lachlan [15] showed that all precomplete universal equivalence relations form one $\sim_{FF}$ degree. The result below relates precomplete equivalence relations with partition graphs. We use the fact that no two distinct equivalence classes of precomplete equivalence relations are recursively separable [16]. Also, recall that we follow the convention that our equivalence relations have infinitely many equivalence classes.

Theorem 22. *Every precomplete equivalence relation realises only the trivial partition graph and is Part-reducible to all other r.e. equivalence relations.*

We would like to observe the way precomplete equivalence relations behave with respect to various reducibilities. From a recursion-theoretic point of view the precomplete relations are the most complex. From an algebraic point of view, however, precomplete equivalence relations behave quite unexpectedly. For instance, as we have already proven, for the class $Isle$ of isles all FF-universal equivalence relations (including the precomplete ones) form the largest $Isle$-degree. In contrast, for the class of partition graphs the precomplete relations belong to the least $Part$-degree. This also stands in contrast to the situation when we consider linear orders [14]; namely, no linear order can be realised over FF-universal equivalence relations.

Here we investigate equivalence relations that realise partition graphs with infinite components. For instance, we show that some FF-universal equivalence relations are more powerful than precomplete ones in terms of partition graphs they realise. We also show some connections between FF-reducibility and $Part$-reducibility.

Theorem 23. *A partition graph $\mathcal{G} = (V, Edge)$ is realised by some FF-universal equivalence relation iff one of the anti-clique components of $\mathcal{G}$ is infinite.*

Theorem 24. *The identity equivalence relation id_ω constitutes the universal Part-degree.*

4 Graphs in general

In this section we want to state the main results which holds for the $Graph$-reducibility based on the class of all graphs and its degrees. Given two r.e. equivalence relations E, E', the definition of $Graph$-reducibility is that $E \leqslant_{Graph} E'$ iff every graph realised by E is also realised by E'.

Theorem 25. **(a)** *There are infinitely many maximal Graph-degrees.*
(b) *There is a least Graph-degree.*
(c) *There are atoms in the Graph-degrees and exactly two of the atoms realise graphs which are not locally finite.*
(d) *There is an ascending chain of type ω which is an initial segment of the graph degrees.*

It is currently open whether in item (c) there are further atoms besides the two which realise graphs which are not locally finite.

References

1. Uri Andrews, Steffen Lempp, Joseph S. Miller, Keng Meng Ng, Luca San Mauro and Andrea Sorbi. Universal computably enumerable equivalence relations. Manuscript, 2012.
2. Walter Baur. Rekursive Algebren mit Kettenbedingungen. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 20:37–46, 1974.
3. Claudio Bernardi. On the relation provable equivalence and on partitions in effectively inseparable sets. *Studia Logica*, 40:29–37, 1981.
4. Claudio Bernardi and Andrea Sorbi. Classifying positive equivalence relations. *The Journal of Symbolic Logic*, 48(3):529–538, 1983.
5. Sam Coskey and Joel David Hamkins. Infinite time computable equivalence relations. *Notre Dame Journal of Formal Logic*, 52(2):203–228, 2011.
6. Sam Coskey, Joel David Hamkins and Russell Miller. The hierarchy of equivalence relations on the natural numbers under computable reducibility. *http://www.arxiv.org/* Report number 1109.3375, 2012.
7. Yuri L. Ershov. Positive equivalence. *Algebra and Logic*, 10(6):378–394, 1974.
8. Yuri L. Ershov. *Theory of numberings.* Nauka, Moscow, 1977 (in Russian).
9. Ekaterina B. Fokina and Sy-David Friedman. Equivalence relations on classes of computable structures. *Fifth Conference on Computability in Europe*, Springer LNCS 5635:198-207, 2009.
10. Ekaterina B. Fokina and Sy-David Friedman. On Σ^1_1 equivalence relations over the natural numbers. *Mathematical Logic Quarterly*, 58(1-2):113-124, 2012.
11. Ekaterina B. Fokina, Sy-David Friedman, Valentina S. Harizanov, Julia F. Knight, Charles F. D. McCoy and Antonio Montalbán: Isomorphism relations on computable structures. *The Journal of Symbolic Logic*, 77(1):122–132, 2012.
12. Ekaterina B. Fokina, Sy-David Friedman and Asger Törnquist. The effective theory of Borel equivalence relations. *Annals of Pure Applied Logic*, 161(7):837–850, 2010.
13. Su Gao and Peter Gerdes. Computably enumerable equivalence relations. *Studia Logica*, 67(1):27–59, 2001.
14. Alexander Gavruskin, Bakhadyr Khoussainov and Frank Stephan. *Reducibilities among equivalence relations induced by recursively enumerable structures.* Manuscript, 2012.
15. Alistair H. Lachlan. A note on positive equivalence relations. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 33:43–46, 1987.
16. Anatoly I. Maltsev. Towards a theory of computable families of objects. *Algebra i Logika*, 3(4):5–31, 1963.
17. Luca San Mauro. *Forma e complessitá. Uno studio dei gradi delle relazioni di equivalenza ricorsivamente enumerabili.* Master's thesis, University of Siena, 2011.
18. Emmy Noether. Idealtheorie in Ringbereichen. *Mathematische Annalen*, 83:24–66, 1921.
19. Piergiorgio Odifreddi. *Classical Recursion Theory.* North-Holland, 1989.
20. Robert I. Soare. *Recursively Enumerable Sets and Degrees.* Springer Verlag, Berlin–New York, 1987.

A Unifying Approach to Decide Timed Relations for Timed Automata and their Game Characterization

Shibashis Guha [1], Shankara Narayanan Krishna [2], Chinmay Narayan [1], S. Arun-Kumar [1]

[1]Indian Institute of Technology Delhi
[2]Indian Institute of Technology Bombay

Abstract. In this paper we present a unifying approach for deciding various bisimulation and simulation equivalences between two timed automata states. We propose a zone based method for deciding these relations in which we eliminate an explicit product construction of the region graphs or the zone graphs as in the classical methods. Our method is also generic and can be used to decide several timed relations. We also present a game characterization for these timed relations and show that the game hierarchy reflects the hierarchy of the timed relations. One can obtain an infinite game hierarchy and thus the game characterization further indicates the possibility of defining new timed relations which have not been studied yet.

1 Introduction

Bisimulation [11] is a widely used relation to assert equivalence of two processes. The relation has been extended for timed systems as well. It is known that timed language equivalence is undecidable for timed automata [2]. Hence for timed automata, bisimulation equivalences are of significant importance since they are known to be decidable [3][1][10][15]. Decidability for timed bisimulation was proved in [3] where a product construction technique on region graph has been used. In [15], the product construction has been applied on zones. We will see that the decidability of time abstracted bisimulation is inherent in the construction of the zone graph we use. We define *corner point bisimulation* and show how checking corner point bisimulation can be used to decide timed bisimulation. In [13], equivalences even weaker than time abstracted bisimulation have been defined. Our zone graph approach can be used to decide these relations as well. Corresponding to every bisimulation relation, a simulation equivalence can be considered. In this paper, we present a unifying approach to decide several of these relations. We also present a game semantics corresponding to these timed relations which is similar to Stirling's bisimulation game for discrete time relations [12]. The game theoretic formulations of these relations seem to free us from the more tedious operational reasoning that is sometimes required to define the relationships between various relations. This generalized game semantics has

certain parameters which on being assigned different values produces a hierarchy of timed games that also reflects the hierarchy of the timed relations. Since it is possible to construct an infinite game hierarchy, it also throws light on the possibility of defining several new timed relations.

Thus the main contribution of this work includes presenting a unifying approach to decide various timed relations and defining the game characterization similar to Stirling's bisimulation games for the timed relations.

In section 2, we give a brief introduction to timed automata, introduce several definitions required in the paper and describe the way we construct the zone graph, which is termed a *zone valuation graph* [8]. Section 3 describes the various timed and time abstracted relations considered in this work. In section 4, we present the methods for deciding various timed relations. The game semantics is given in section 5. Finally, we conclude in section 6 with the emphasis that zone valuation graph can be used as a common framework to decide several kinds of timed relations.

2 Timed Automata

Timed automata [2] is an approach to modelling time critical systems where the system is modeled with *clocks* that track elapsed time. It is a finite-state structure that can manipulate real-valued clock variables. Corresponding to every transition, a subset of the clocks can be specified that can be *reset* to zero. Clock constraints also specify the condition for actions being enabled. If the constraints are not satisfied, the actions will be disabled. The clock constraints $\mathcal{B}(C)$ over a set of clocks C is given by the grammar $g ::= \ x \backsim c \mid g \wedge g$, where $c \in \mathbb{N}$ and $x \in C$ and $\backsim \in \ \{\leq, <, =, >, \geq\}$. A *timed automaton* over a finite set of clocks C and a finite set of actions Act is a quadruple (L, l_0, E, C) where L is a finite set of locations, ranged over by l, $l_0 \in L$ is the initial location, $E \subseteq L \times \mathcal{B}(C) \times Act \times 2^C \times L$ is a finite set of *edges*.

2.1 Semantics

The semantics of a timed automaton can be described with a *timed labeled transition system*(TLTS)[1]. Let $A = \ (L, l_0, E, C)$ be a timed automaton over a set of clocks C and a set of visible actions Act. The timed transition system $T(A)$ generated by A can be defined as $T(A) = (Proc, Lab, \{\xrightarrow{\alpha} \mid \alpha \in Lab\})$, where $Proc = \{(l, v) \mid (l, v) \in L \times (C \to \mathbb{R}_{\geq 0})$, i.e. *states* are of the form (l, v), where l is a location of the timed automaton and v is a valuation assigned to the clocks of A. $Lab = Act \cup \mathbb{R}_{\geq 0}$ is the set of labels; and the transition relation is defined by $(l, v) \xrightarrow{a} (l', v')$ if for an edge $(l \xrightarrow{g,a,r} l') \in \ E$, $v \models g, v' = v_{[r \leftarrow 0]}$, where the edge $(l \xrightarrow{g,a,r} l')$ denotes that l is the source location, g is the guard, a is the action, r is the set of clocks to be reset and l' is the target location. $(l, v) \xrightarrow{d} (l, v + d)$ for $d \in \mathbb{R}_{\geq 0}$ and $v + d$ is the valuation in which every clock value is incremented by d. Let v_0 denote the valuation such that $v_0(x) = 0$ for all $x \in C$. (l_0, v_0) is the initial state of $T(A)$.

We now define various concepts that will be used in the paper.

Definition 1. ***sort:*** *For a timed automata state p, $sort(p)$ denotes the set of visible actions that can be performed by p. So $sort(p) \subseteq Act$.*

Definition 2. *Timed trace: If p' is a timed state that is reachable from the initial state p, then p' can be reached from p by a sequence of delays and actions as follows: $p \xrightarrow{d_1} p_1 \xrightarrow{a_1} p'_1 \xrightarrow{d_1} p_2 \xrightarrow{a_1} p'_2 \cdots \xrightarrow{d_n} p_n \xrightarrow{a_n} p'$, where this sequence of delays and actions to reach p' is termed as a timed trace.*

Definition 3. ***zone:*** *A zone z is a set of valuations defined by a conjunction of two kinds of clock constraints: for $x, y \in C$, the set of clocks of a timed automaton, $x \backsim c$ or $x - y \backsim c$, where $c \in \mathbb{Z}$.*

In a *zone graph*, a node is a tuple of location and a zone and the edges are the transition relations between these nodes defined as follows. $(l, z) \xrightarrow{a} (l', z')$, where $a \in Act$, if for every v satisfying z, $\exists v'$ satisfying z' such that $(l, v) \xrightarrow{a} (l', v')$. If the zones corresponding to (l, v) and (l, v') be z and z' respectively and there is a transition in $T(A)$ such that $(l, v) \xrightarrow{d} (l, v')$, then we have an edge $(l, z) \xrightarrow{\varepsilon} (l, z')$ in the zone graph. Every node has an ε transition to itself and the ε transitions are also transitive in nature. For both a and ε transitions, if z is a zone then z' is also a zone. We denote a node of the zone graph with s. A zone graph can be formally defined using the quadruple $(S, s_0, Lep, \rightarrow)$, where S is the set of nodes of the zone graph, s_0 is the initial node, $Lep = Act \cup \{\varepsilon\}$ and $\rightarrow$ denotes the set of transitions.

Definition 4. *Pre-stability: A zone z_1 is forward stable or pre-stable with respect to another zone z_2 if $z_1 \subseteq preds(z_2)$ or $z_1 \cap preds(z_2) = \emptyset$ where $preds(z) \stackrel{def}{=} \{v \in V | \exists v' \models z$ such that $v \xrightarrow{\alpha} v'$ where $\alpha \in Act \cup \mathbb{R}_{\geq 0}\}$, V being the possible set of clock valuations.*

A zone valuation graph $Z_{(A,p)}$ corresponds to a particular state p of the timed automaton A. The clock valuation of p is same as the initial clock valuation corresponding to which the zone valuation graph is created. For a state $q \in T(A)$, $\mathcal{N}(q)$ represents the node of the zone valuation graph with the same location as that of q and whose clock valuation range includes the valuation of q. For two zone valuation graphs, $Z_{(A_1,p)} = (S_1, s_p, Lep, \rightarrow_1)$, $Z_{(A_2,q)} = (S_2, s_q, Lep, \rightarrow_2)$ and a relation $\mathcal{R} \subseteq S_1 \times S_2$, $Z_{(A_1,p)}\, \mathcal{R}\, Z_{(A_2,q)}$ iff $(s_p, s_q) \in \mathcal{R}$. While checking $\mathcal{R}$, ε is considered visible similar to an action in Act. An ε action represents a delay $d \in \mathbb{R}_{\geq 0}$, where $d \geq 0$. The detailed algorithm for creating zone valuation graph has been described in algorithm 1 and consists of forward analysis of the timed automaton. The set of valuations for every location is initially split into zones based on the canonical decomposition of its outgoing transition as discussed in [13]. The forward analysis may cause a zone graph to become infinite [6]. Several kinds of abstractions have been proposed in the literature [5][6][7]. We use *location dependent maximal constants* abstraction [6] to be used for creating the finite zone valuation graph. The algorithm has a splitting phase (phase 1)

Algorithm 1 Construction of Zone Valuation Graph
Input: Timed automaton A
Output: Zone valuation graph corresponding to A

1: Calculate max_x^l for each location $l \in L$ and each clock $x \in C$. This is required for abstraction to ensure finite number of zones in the zone graph.
2: Initialize Q to an empty queue.
3: $Enqueue(Q, < l_0, \emptyset >)$. /* Every element is a tuple of a location and its parent */
4: successors_added = false.
5: **while** Q not empty **do**
6: $< l, l_p >= dequeue(Q)$
7: **if** $l_p \neq \emptyset$ **then**,
8: For the edge $l_p \xrightarrow{g,a,X'} l$ in A, for each existing zone z_{l_p} of l_p, create the zone $z = (z_{l_p} \uparrow \cap g_{[X' \leftarrow \overline{0}]}) \uparrow$ of l, when $z \neq \emptyset$.
9: Abstract each of the newly created zones if necessary and for any newly created zone z, for location l, if $\exists z_1$ of same location such that $z \cap z_1 \neq \emptyset$, then merge z and z_1.
10: Update edges from zones of l_p to zones of l appropriately.
11: If a new zone of l is added or an existing zone of l is modified, then for all successors l_j of l, enqueue $< l_j, l >$ to Q.
12: successors_added = true.
13: **end if**
14: Find the canonical decomposition of constraint based on the guards of the outgoing transitions of l.
15: Split the existing zones of l further based on this canonical decomposition mentioned above. Note that the zones created from this split are convex.
16: Abstract each of the newly created zones if necessary.
17: Update edges appropriately.
18: **if** any new zones of l are created or any existing zones of l are modified due to the canonical decomposition of the outgoing edges of l and successors_added = false **then**
19: for all the successor locations l_j of l to Q, enqueue $< l_j, l >$ to Q.
20: **end if**
21: **end while**
22:
23: /* Phase 2 : In this phase, pre stability is enforced */
24: $new\_zone = 1$
25: **while** $new\_zone = 1$ **do**
26: $new\_zone = 0$
27: **for all** edges $l_i \xrightarrow{g,a,X'} l_j$ **do**
28: **for all** pairs of zones z_{lik}, z_{ljm} such that $z_{lik} \xrightarrow{a} z_{ljm}$ is an edge in the zone graph **do**
29: **if** $!(z_{lik} \not\subseteq [z_{ljm\downarrow_{[X' \leftarrow \overline{0}]^{-1}}} \cap g]\downarrow)$ **then**
30: Split z_{lik} into $[z_{ljm\downarrow_{[X' \leftarrow \overline{0}]^{-1}}} \cap g]\downarrow$ and $z_{lik} - [z_{ljm\downarrow_{[X' \leftarrow \overline{0}]^{-1}}} \cap g]\downarrow$ /* Note that this split still maintains convexity of z_{lik} since the zone is split entirely along an axis that is parallel to the diagonal in the n-dimensional space. */
31: $new\_zone = 1$
32: Update the edges
33: **end if**
34: **end for**
35: **end for**
36: **end while**

followed by a phase 2 in which pre-stability is ensured for every transition. Pre-stability implies that for some state (l, v) in a node (l, z) such that for a $v \models z$, for a timed trace tr, if $(l, v) \xrightarrow{tr} (l'', v'')$, where (l'', v'') is a state of node (l'', z''), then $\forall v' \models z$, $\exists tr'.(l, v') \xrightarrow{tr'} (l'', \tilde{v})$, such that $\hat{tr'} = \hat{tr}$ and $\tilde{v} \models z''$ and hence $(l'', \tilde{v})$ belongs to the zone (l'', z''). Here $\hat{tr}$ represents the sequence of visible actions in tr. According to the construction given in algorithm 1 , for a given location of the timed automaton, the zones corresponding to any two nodes are disjoint.

3 Equivalences for Timed Systems

In this section, we define the timed and the time abstracted relations considered in this work. We only consider the strong form of these relations here. We enumerate a few clauses first using which we define $p_1 \mathcal{R} p_2$ where p_1 and p_2 are two timed automata states and $\mathcal{R}$ is a timed or a time abstracted relation.

1. $\forall a \in Act \wedge \forall p_1', p_1 \xrightarrow{a} p_1' \Rightarrow [\, \exists p_2' : p_2 \xrightarrow{a} p_2' \wedge p_1' \mathcal{R} p_2' \,]$
2. $\forall a \in Act \wedge \forall p_1', p_1 \xrightarrow{a} p_1' \Rightarrow [\, \exists p_2' \, \exists d' \in \mathbb{R}_{\geq 0} : p_2 \xrightarrow{d'} \xrightarrow{a} p_2' \wedge p_1' \mathcal{R} p_2' \,]$
3. $\forall a \in Act \wedge \forall p_1', p_1 \xrightarrow{a} p_1' \Rightarrow [\, \exists p_2' \, \exists d_1, d_2 \in \mathbb{R}_{\geq 0} : p_2 \xrightarrow{d_1} \xrightarrow{a} \xrightarrow{d_2} p_2' \wedge p_1' \mathcal{R} p_2' \,]$
4. $\forall d \in \mathbb{R}_{\geq 0} \wedge \forall p_1', p_1 \xrightarrow{d} p_1' \Rightarrow [\, \exists p_2' : p_2 \xrightarrow{d} p_2' \wedge p_1' \mathcal{R} p_2' \,]$
5. $\forall d \in \mathbb{R}_{\geq 0} \wedge \forall p_1', p_1 \xrightarrow{d} p_1' \Rightarrow [\, \exists p_2' \, \exists d' \in \mathbb{R}_{\geq 0} : p_2 \xrightarrow{d'} p_2' \wedge p_1' \mathcal{R} p_2' \,]$

p_2 time simulates p_1 if the clauses 1 and 4 hold. $\mathcal{R}$ is a *timed simulation equivalence* if p_1 time simulates p_2 and p_2 time simulates p_1. A symmetric timed simulation is a *timed bisimulation* relation. A symmetric relation that satisfies clauses 1 and 5 is a *time abstracted bisimulation.* A relation that is symmetric and satisfies clauses 2 and 5 is a *time abstracted delay bisimulation relation.* A symmetric relation satisfying clauses 3 and 5 is a *time abstracted observational bisimulation.*

The corresponding largest bisimulation relations are called *bisimilarity* relations and they are *timed bisimilarity* ($\sim_t$), *time abstracted bisimilarity* ($\sim_u$), *time abstracted delay bisimilarity* ($\sim_y$), *time abstracted observational bisimilarity* ($\sim_o$). It is easy to see from the definitions that strong timed bisimulation implies strong time-abstracted bisimulation whereas the converse is not true. Besides, the definitions imply $\sim_u \subseteq \sim_y \subseteq \sim_o$. Also the existence of a bisimulation relation between two states implies the existence of the corresponding simulation equivalence. Hence we have $\sim_t \subseteq \sim_u \subseteq \sim_y \subseteq \sim_o$ and similar containment relations also exist among the corresponding simulation equivalences.

4 Deciding relations for Timed Automata

In this section, we present a unifying approach to decide several relations for timed automata using zone valuation graph.

4.1 Deciding Timed Bisimulation

In timed bisimulation, the delay made by one state has to be matched exactly by the other state. Timed bisimulation has been proved to be decidable for timed automata in [3] where a product construction technique on the region graphs has been used. In [15], product construction is applied on zone graph instead of region graph for deciding timed bisimulation. In order to decide this relation, we define *corner point bisimulation* relation and argue that corner point bisimulation relation is decidable for timed automata. Then we prove that the corner point bisimulation actually coincides with timed bisimulation. Our method eliminates the product construction required on zone graphs. Timed bisimulation has infinitely many equivalent classes. However, we show that instead of checking all possible delays, timed bisimulation for timed automata can be decided by checking delays of the form n, $n+\delta$ or $n-\delta$, where $n \in \{0, 1, \ldots, C\}$, where $C = max(C_A, C_B)$, C_A and C_B being the maximum constants with which any of the clocks has been compared in timed automata A and B respectively. We define corner point bisimulation formally below.

Definition 5. *Corner point simulation (cp-simulation): A relation $\mathcal{R}$ is a corner point simulation relation, if for two timed automata states p and q, $(p,q) \in \mathcal{R}$, i.e. q simulates p and the following conditions hold.*
For every visible action $a \in Act$, if $p \xrightarrow{a} p'$, then $\exists q'$ such that $q \xrightarrow{a} q'$ and $p'\mathcal{R}q'$
If p is in node $\mathcal{N}_p$, consider the maximum possible delay d from p such that $p \xrightarrow{d} p'$ and p' is in node $\mathcal{N}_p$, $\exists q'$ such that $q \xrightarrow{d} q'$ and $p'\mathcal{R}q'$
For every node $\mathcal{N}_{p'} \neq \mathcal{N}_p$ such that $\mathcal{N}_p \xrightarrow{\varepsilon} \mathcal{N}_{p'}$, consider the minimum delay d from p such that $p \xrightarrow{d} p'$ and p' is in node $\mathcal{N}_{p'}$, $\exists q'$ such that $q \xrightarrow{d} q'$ and $p'\mathcal{R}q'$

A symmetric corner point simulation relation is a corner point bisimulation (cp-bisimulation).

Definition 6. *Corner point trace: Consider a pair of states (p', q') appearing in a corner point bisimulation relation. A timed trace from the state p to the corner point p' following the delays and actions specified in definition 5 is termed a corner point trace.*

Lemma 1. *For the timed automata states p and q, there are only finitely many pairs of states in $\mathcal{R}$ where $\mathcal{R}$ is a corner point simulation or bisimulation relation.*

Theorem 1. *Corner point simulation and corner point bisimulation relations are decidable.*

Lemma 2. *For two timed automata initial states p and q, $p \sim_t q \Rightarrow p\mathcal{R}q$.*

We will now prove that corner point bisimulation between two timed automata states implies that they are timed bisimilar.

Fact 1 *If p and q are not timed bisimilar, then one of the following conditions hold true.*

– *There exists a timed trace* tr *such that* $p \xrightarrow{tr} p'$ *and* $\forall q'$ *such that* $q \xrightarrow{tr} q'$ *and* $sort(p') \neq sort(q')$.
– *There exists a timed trace* tr *such that* $q \xrightarrow{tr} q'$ *and* $\forall p'$ *such that* $p \xrightarrow{tr} p'$ *and* $sort(q') \neq sort(p')$.

Taking contrapositive of the above fact, if for every timed trace tr from p, such that $p \xrightarrow{tr} p'$, $\exists q'$ such that $q \xrightarrow{tr} q'$ and $sort(p') = sort(q')$ and similarly for every timed trace tr from q, such that $q \xrightarrow{tr} q'$, there exists a p' such that $p \xrightarrow{tr} p'$ and $sort(q') = sort(p')$, then p and q are timed bisimilar.

We prove that if p and q are cp-bisimilar, then the above condition holds. We state this formally below. In the statement of the lemma, we abuse the notation $p_1 + d_1$ to denote a state where all clock valuations of p_1 have been increased by d_1.

Lemma 3. *If* p_1 *and* q_1 *are cp-bisimilar, then the following conditions hold true for all* $n \in \mathbb{N}$.

– *For all delays* $d_1, d_2, \ldots, d_n \in \mathbb{R}_{\geq 0}$ *and* $\forall a_1 \in sort(p_1 + d_1)$, $\forall a_2 \in sort(p_2 + d_2), \ldots,$ $\forall a_n \in sort(p_n + d_n)$ *such that* $p_1 \xrightarrow{d_1}\xrightarrow{a_1} p_2 \xrightarrow{d_2}\xrightarrow{a_2} \cdots p_n \xrightarrow{d_n}\xrightarrow{a_n} p_{n+1}$, $\exists a_1 \in sort(q_1+d_1)$, $\exists a_2 \in sort(q_2+d_2)$, $\ldots$, $\exists a_n \in sort(q_n+d_n)$, *such that* $q_1 \xrightarrow{d_1}\xrightarrow{a_1} q_2 \xrightarrow{d_2}\xrightarrow{a_2} \cdots q_n \xrightarrow{d_n}\xrightarrow{a_n} q_{n+1}$ *and* p_{n+1} *and* q_{n+1} *can perform the same set of actions after same delay* $d_{n+1} \in \mathbb{R}_{\geq 0}$, *i.e.* $sort(p_{n+1}+d_{n+1}) = sort(q_{n+1} + d_{n+1})$.
– *For all delays* $d_1, d_2, \ldots, d_n \in \mathbb{R}_{\geq 0}$ *and* $\forall a_1 \in sort(q_1 + d_1)$, $\forall a_2 \in sort(q_2 + d_2), \ldots,$ $\forall a_n \in sort(q_n + d_n)$ *such that* $q_1 \xrightarrow{d_1}\xrightarrow{a_1} q_2 \xrightarrow{d_2}\xrightarrow{a_2} \cdots q_n \xrightarrow{d_n}\xrightarrow{a_n} q_{n+1}$, $\exists a_1 \in sort(p_1+d_1)$, $\exists a_2 \in sort(p_2+d_2)$, $\ldots$, $\exists a_n \in sort(p_n+d_n)$, *such that* $p_1 \xrightarrow{d_1}\xrightarrow{a_1} p_2 \xrightarrow{d_2}\xrightarrow{a_2} \cdots p_n \xrightarrow{d_n}\xrightarrow{a_n} p_{n+1}$ *and* p_{n+1} *and* q_{n+1} *can perform the same set of actions after same delay* $d_{n+1} \in \mathbb{R}_{\geq 0}$, *i.e.* $sort(p_{n+1}+d_{n+1}) = sort(q_{n+1} + d_{n+1})$.
– *If* $p \xrightarrow{tr} p'$ *and* $q \xrightarrow{tr} q'$ *where* $\mathcal{N}p'$ *and* $\mathcal{N}q'$ *are the nodes containing timed states* p' *and* q' *respectively, then there exists at least one corner point trace* tr_i *such that* $\hat{tr} = \hat{tr_i}$ *and* $p \xrightarrow{tr_i} \tilde{p}$ *and* $q \xrightarrow{tr_i} \tilde{q}$ *such that* $\tilde{p}$ *is a state in the node* $\mathcal{N}_{p'}$ *and* $\tilde{q}$ *is a state in the node* $\mathcal{N}_{q'}$.

Lemma 4. *For two timed automata initial states* p *and* q, $p\mathcal{R}q \Rightarrow p \sim_t q$, *where* $\mathcal{R}$ *is a corner point bisimulation relation.*

We state here the main theorem of the paper for deciding timed bisimulation by deciding cp-bisimulation by combining the lemma 2 and 4.

Theorem 2. *For two timed automata states* p *and* q, p *and* q *are timed bisimilar if and only if* p *and* q *are cp-bisimilar.*

4.2 Deciding Time Abstracted Bisimulation

Time abstracted bisimulation has been shown to be decidable [1][10] using region graph [2]. For the two given timed states, their region graphs are constructed. If the region graphs are strongly bisimilar, then the two states are time abstracted bisimilar. We use zone valuation graph instead of region graph. The size of zone valuation graph, is independent of the constants with which the clocks are compared in the timed automaton guards. Let $Z_{(A_1,p)}$ be the zone graph for timed state p of timed automaton A_1. If there are two valuations (l, v) and (l, v') such that they belong to the same node, then by construction of $Z_{(A_1,p)}$, (l, v) and (l, v') are time abstracted bisimilar. Thus in the zone graph, it is the case that a state (l, v) in the TLTS of A is time abstracted bisimilar to the zone z in the zone graph $Z_{(A_1,p)}$. The same holds for a timed state (l_2, v_2) of the TLTS corresponding to another timed automaton A_2. Thus checking whether two states (l_1, v_1) and (l_2, v_2) of two timed automata A_1 and A_2 are time abstracted bisimilar involves checking whether their corresponding nodes in the two timed automata are strongly bisimilar. The following theorems show how time abstracted delay bisimulation and time abstracted observational bisimulation [13] too can be decided along with strong time abstracted bisimulation using zone valuation graph.

Theorem 3. *Let $\mathcal{R} \subseteq S_1 \times S_2$ be a symmetric relation. Two nodes $(s_1, s_2) \in \mathcal{R}$ if and only if $\forall a \in Act, \forall s_1'[s_1 \xrightarrow{a} s_1' \Rightarrow \exists s_2' \,.\, s_2 \xrightarrow{\beta} s_2'$ and $(s_1', s_2') \in \mathcal{R}]$ and $\forall s_1', [s_1 \xrightarrow{\varepsilon} s_1' \Rightarrow \exists s_2' \,.\, s_2 \xrightarrow{\varepsilon} s_2'$ and $(s_1', s_2') \in \mathcal{R}]$*
Two states p and q are strong timed bisimilar iff $Z_{A_1,p}\ \mathcal{R}\ Z_{A_2,q}$ and β is the transition a, they are time abstracted delay bisimilar if β is the sequence of transitions $\xrightarrow{\varepsilon}\xrightarrow{a}$ whereas p and q are time abstracted observational bisimilar if β is $\xrightarrow{\varepsilon}\xrightarrow{a}\xrightarrow{\varepsilon}$.

4.3 Complexity of Deciding Bisimulation Relations

In our work, we decide the timed and the time abstracted relations using a zone graph approach. For a given location, the zones in the zone graph are disjoint. Thus the size of the zone graph is limited by the size of the region graph and it is thus exponential in the number of clocks of the timed automaton. However, in most cases, the size of the zone graph is smaller than the size of the region graph. Besides, for checking timed bisimulation, a product construction on the region graphs characterizes the common behaviour of the two timed automata and one still needs to store the original region graphs as well in order to check for timed simulation relation or timed bisimulation. The same argument holds for the product construction on the zone graphs as well used in [3]. Thus since we do not use a product construction on the zones and use the individual zone valuation graphs of the two timed automata directly for deciding the relations, our algorithm uses less space. Deciding timed bisimulation and timed simulation is known to be EXPTIME-complete [9]. Thus our algorithm is not asymptotically better than existing approaches, however it has better running time and uses lesser space than existing approaches.

5 Game Characterization

Bisimulation games were defined in [12] for discrete processes. In [4], bisimulation game has been extended for the Van Glabbeek's spectrum [14]. We present here game characterizations for timed relations that is similar to bisimulation games and define the game semantics using zone valuation graph. As in the bisimulation game, the game is played in rounds on two graphs. The game may be played between the nodes of the zone valuation graph (game for time abstracted relations) or between the timed states appearing in some node of the zone valuation graph (game for timed relations). In each round, the challenger chooses a graph and the defender tries to make a corresponding move on the other graph where the correspondence of the moves is defined in the following subsection in terms of the ordered tuple α. If the defender can always make a move in response to the challenger's move, then it has a winning strategy implying that the two states are related through the relation that corresponds to the game. Otherwise it loses which implies that the two states are not related in which case the challenger is said to have a winning strategy. If the challenger changes the graph between two consecutive rounds, it is known as an *alternation*. Alternations are not allowed in simulation preorder and simulation equivalence games. A game always *terminates* due to the finiteness of the zone valuation graph. In the games described in this section, the moves denote a visible action or a delay action or a sequence of actions belonging to the set $Act \cup \{\varepsilon\}$.

5.1 Game Template

A timed game proposed in this work can be described as $n - \Gamma_k^\alpha$. Each game is characterized by the following parameters:

- n : number of alternations. If not mentioned, then there is no restriction on the number of alternations.
- $k \in \{\mathbb{N} \cup \infty\}$: number of rounds; $n \leq k-1$ when $k \neq \infty$.
- α : an ordered tuple $\langle \alpha_1, \alpha_2 \rangle$. α_1 denotes the move chosen by the challenger. Depending on the game for the timed relation, either $\alpha_1 \in Lep$ or $\alpha_1 \in Act \cup \mathbb{R}_{\geq 0}$ whereas α_2 denotes the move chosen by the defender and may be the same as α_1 or may be a sequence of the form $\varepsilon.\alpha_1$ or $\varepsilon.\alpha_1.\varepsilon$, e.g. for the pair $\langle a/\varepsilon, \varepsilon.a/\varepsilon \rangle$ where $a \in Act$, the challenger makes a move a whereas the defender's move consists of ε followed by an a. Particularly, in the timed bisimulation game, α is assigned $\langle a/d, a/d \rangle$, which denotes that a visible action by the challenger has to be matched by the defender and a delay action d by the challenger has to be matched with an exact delay d move by the defender.

5.2 Hierarchy of Games

A hierarchy among the timed relations discussed in this paper is captured in figure 1(a). We show here several lemmata which capture this hierarchy through

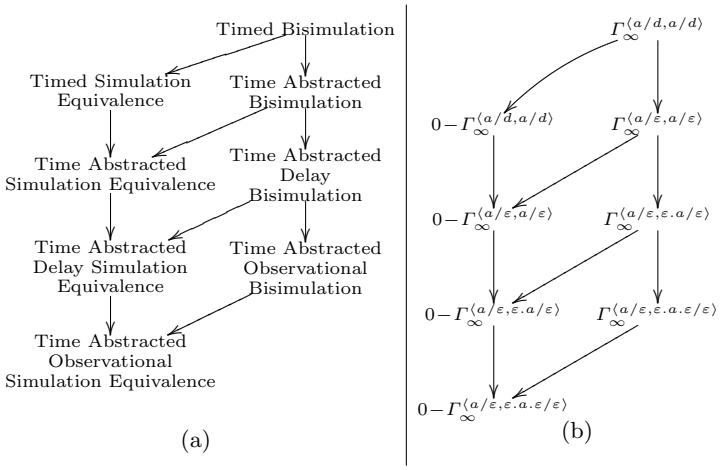


Fig. 1. (a) presents the spectrum of timed relations and (b) shows timed games corresponding to these relations

the game semantics. These lemmata also help us build an infinite game hierarchy which also suggests defining several new timed relations that do not exist in the literature. The arrow from a game Γ_1 to a game Γ_2 denotes that if the defender has a winning strategy for Γ_1, then it also has a winning strategy for Γ_2. Besides in each of the following lemmas, for each pair of games, if $\Gamma_1 \longrightarrow \Gamma_2$, then $\Gamma_2 \not\longrightarrow \Gamma_1$. Figure 1(b) shows the games corresponding to the relations shown in figure 1(a). The game hierarchy reflects the hierarchy of the timed relations.

Lemma 5. $\Gamma_\infty^\alpha \longrightarrow n-\Gamma_\infty^\alpha \longrightarrow (n-1)-\Gamma_\infty^\alpha$, *for all* $n > 0$
$\Gamma_k^\alpha \longrightarrow n-\Gamma_k^\alpha \longrightarrow (n-1)-\Gamma_k^\alpha$, *for all* $k > 0$, $n < k$

Other parameters remaining the same, if the defender has a winning strategy when the challenger is allowed more alternations, then the defender will also win the game where the challenger is allowed lesser number of alternations.

Lemma 6. $\Gamma_\infty^\alpha \longrightarrow \Gamma_k^\alpha \longrightarrow \Gamma_{k-1}^\alpha$, *for all* $k > 0$
$n-\Gamma_\infty^\alpha \longrightarrow n-\Gamma_k^\alpha \longrightarrow n-\Gamma_{k-1}^\alpha$, *for all* $k > 0$, $n < k$

Other parameters remaining the same, if the defender wins the game with more number of rounds, then it also wins the game which has lesser number of rounds in the game.

Lemma 7. $n - \Gamma_k^{\langle a/d,a/d\rangle} \longrightarrow n - \Gamma_k^{\langle a/\varepsilon,a/\varepsilon\rangle} \longrightarrow n - \Gamma_k^{\langle a/\varepsilon,\varepsilon.a/\varepsilon\rangle} \longrightarrow n - \Gamma_k^{\langle a/\varepsilon,\varepsilon.a.\varepsilon/\varepsilon\rangle}$

If the defender can match a delay action exactly as in the corner point bisimulation, then it can match an epsilon move of the challenger. Also if the defender can reply to a visible action of the challenger, then it can reply with an $\varepsilon.a$ or an $\varepsilon.a.\varepsilon$ move since ε may represent a zero delay.

6 Conclusion

In this paper, we present a unified zone based approach to decide various timed relations between two timed automata states. In our method, we do not need explicit product construction of regions or zones for deciding these relations as used in [3] or [15]. We also provide a game semantics for deciding these timed relations and show that the hierarchy among the games reflects the hierarchy among the timed relations. The advantage of a game-theoretic formulation is that it allows fairly general relationships between the parameters on Γ to define the hierarchy. The fine-tuning and variations of these parameters allow formulations of many more equivalences and preorders than the ones present in the literature related to behavioural equivalences involving real time which otherwise may not be easily captured through operational definitions and reasoning.

References

1. L. Aceto, A. Ingólfsdóttir, K.G. Larsen, and J. Srba. *Reactive Systems: Modelling, Specification and Verification*. Cambridge University Press, 2007.
2. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. K. Cerans. Decidability of bisimulation equivalences for parallel timer processes. In *Proceedings of CAV*, volume 663, pages 302–315. Springer-Verlag, 1992.
4. X. Chen and Y. Deng. Game characterizations of process equivalences. In *Proceedings of APLAS*, pages 107–121, 2008.
5. C. Daws and S. Tripakis. Model checking of real-time reachability properties using abstractions. In *Proceedings of TACAS*, pages 313–329. Springer-Verlag, 1998.
6. E. Fleury G. Behrmann, P. Bouyer and K. G. Larsen. Static guard analysis in timed automata verification. In *Proceedings of TACAS*, pages 254–270. Springer-Verlag, 2003.
7. K. G. Larsen G. Behrmann, P. Bouyer and R. Pelanek. Lower and upper bounds in zone-based abstractions of timed automata. *International Journal on Software Tools for Technology Transfer*, 8:204–215, 2006.
8. S. Guha, C. Narayan, and S. Arun-Kumar. On decidability of prebisimulation for timed automata. In *Proceedings of CAV*, pages 444–461. Springer-Verlag, 2012.
9. F. Laroussinie and Ph. Schnoebelen. The state explosion problem from trace to bisimulation equivalence. In *Proceedings of FoSSaCS*, pages 192–207. Springer-Verlag, 2000.
10. K. G. Larsen and W. Yi. Time abstracted bisimulation: implicit specifications and decidability. In *Proceedings of MFPS*, volume 802, pages 160–176. Springer-Verlag, 1994.
11. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
12. C. Stirling. Local model checking games. In *Proceedings of CONCUR*, pages 1–11, 1995.
13. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18:25–68, 2001.
14. Rob J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In *Proceedings of CONCUR*, pages 278–297, 1990.
15. C. Weise and D. Lenzkes. Efficient scaling-invariant checking of timed bisimulation. In *Proceedings of STACS*, volume 1200, pages 177–188. Springer, Berlin, 1997.

Vector Addition Systems With Split/Join Transitions: A Covering Theorem

Paulin Jacobé de Naurois

CNRS, UMR 7030, Laboratoire d'Informatique de Paris-Nord (LIPN), Université Paris 13, Sorbonne Paris Cité, F-93430, Villetaneuse, France.

Abstract. We introduce Vector Addition Systems with Split/Join transitions (VASS-SJ), a symmetric extension of Branching VASS. We introduce a suitable notion of covering tree (Karp and Miller Tree) for the model, and prove its finiteness and effective constructibility. We use this covering tree to obtain a coverability result, a key step-stone towards a reachability algorithm.

Introduction

Petri Nets (PNs), and equivalent models of computation such as Vector Addition Systems (VAS) and Vector Addition Systems with States (VASS) are a natural resource sensitive model of computation, for which reachability (for PNs) or, equivalently, positive reachability (for VASs and VASSs), among other key properties, has been proven decidable [Kos82,May84,May81,Lam92,Reu88]. Several natural extensions of this model have been studied (e.g colored, hierarchical, prioritized, recursive PNs) for which reachability has been proven either decidable or undecidable.

Another natural extension of PNs are Vector Addition Tree Automata (VATA), introduced by de Groote, Guillaume and Salvati [dGGS04], who prove the equivalence between reachability in the VATA model and provability in Multiplicative Exponential Linear Logic (MELL), a problem whose decidability status is still unknown today. VATA behave like Petri nets enriched with a "join" transition, where two distinct markings are merged together in a new one. The reachability problem for VATA is then whether a given set of initial markings can yield a given final marking by a finite sequence of transition firings.

Independently to the former authors, Verma and Goubault-Larrecq [VGL05] introduced a branching extension of VASS, called BVASS, for which they proposed a notion of Karp and Miller Tree that allows one to establish the decidability of properties such as finiteness, boundedness and emptiness of their model. Yet, they do not obtain the decidability or undecidability of the reachability problem. It turns out that the BVASS model is actually equivalent to the VATA model of de Groote, Guillaume and Salvati, and the positive reachability of the former is equivalent to the reachability of the latter. Further work on the complexity of decision problems for the BVASS model has been done by Demri et al. [DJLL09] and by Lazic [Laz10].

A careful reading of the reachability algorithm of Kosaraju and Mayr [Kos82], [May84,May81] for PNs or for VASS positive reachability reveals a feature of the model that is central in the construction of the algorithm: PNs, as well as its equivalent models, VAS and VASS, are symmetric, in the sense that the model is stable by an inversion of the arrows. This feature is not present in the VATA or in the BVASS model, hence it seems unlikely that one can adapt the algorithm of Kobayashi and Mayr to the setting of VATA or BVASS.
Our contribution to this line of research consists in the symmetrization of the BVASS model, that we call Vector Addition Systems with States and Split/Join transitions (VASS-SJ). VASS-SJs are not functional rewriting systems, and, as such, are not subject to the study of Finkel and Goubault-Larrecq [FG12]. As a first step towards a reachability result for VASS-SJ, we also introduce a suitable extension of the classical notion of Karp and Miller Tree for VASS to this new model of VASS-SJ. This extension is a pair of two elements: a directed, acyclic graph, and a partition of its set of vertices that quotients it to a tree. We prove its finiteness and effective constructibility, and prove its use for coverability results: we prove that the generalized configurations of the Karp and Miller tree capture in a synthetic way all unbounded configurations that are positively reachable. This result, Theorem 3, can be seen as a key step-stone towards a generalization of the classical reachability algorithm for PNs to PN-SJ.
The first section is devoted to the definition of the model, and a first reachability result, the decidability of its $\mathbb{Z}$-reachability. The Karp and Miller tree is introduced in Section 2, where its finiteness and constructibility is stated. This Karp and Miller tree is used in Section 3 for a first $\mathbb{N}$-coverability result. The proofs, too long for a short presentation, are postponed to a later publication.

1 VASS-SJ

1.1 Definitions

Definition 1. *VASS-SJ.*
A Vector Addition Systems with States and Split/Join transitions (VASS-SJ) is a 4-tuple $S = (G, T, m, v)$, where:

- $G = (Q, A)$ *is a finite directed graph, whose vertices are called* states*,*
- $T \subseteq A \cup A^2$ *is a set of* transitions*,*
- $m \geq 1$ *is a natural number,*
- $v : T \to \mathbb{Z}^m \cup \{s\} \cup \{j\}$ *is a* value *function, such that*
  regular transition: $v(t) \in \mathbb{Z}^m$ *if and only if* $t \in A$*,*
  split transition: $v(t) = s$ *if and only if* $t = (a_1, a_2)$ *and* a_1 *and* a_2 *share the same origin,*
  join transition: $v(t) = j$ *if and only if* $t = (a_1, a_2)$ *and* a_1 *and* a_2 *share the same destination.*

Definition 2. *Configuration of a VASS-SJ.*
Let $S = (G, T, m, v)$ be a VASS-SJ. A single configuration *of G is a 2-tuple*

$c = (q, x)$, *where* $q \in Q$ *is a* state *and* $x \in \mathbb{Z}^m$ *is a* value. *A single configuration* c *is* positive *if and only if no coordinate of its value is negative. A* configuration *of* S *is a multiset* $\mathcal{C}$ *of single configurations of* S. *A configuration is* positive *if and only if all its single configurations are positive.*

1.2 Execution Semantics of a VASS-SJ

Definition 3. *Firing of a Transition.*
Let $S = (G, T, m, v)$ *be a VASS-SJ, and* $\mathcal{C}$ *be a configuration of* S. *Let* t *be a transition of* S. *The* firing *of* t *in* $(S, \mathcal{C})$ *is the relation* $(S, \mathcal{C}) \rightarrow_t (S, \mathcal{C}')$, *where*

if t **is regular:** $t = (q_0, q_1)$, *there exists* $c = (q_0, x_0) \in \mathcal{C}$, *and* $\mathcal{C}' = \mathcal{C} \setminus \{c\} \uplus \{(q_1, x_0 + v(t))\}$.
if t **is split:** $t = ((q_0, q_1), (q_0, q_1'))$, *there exists* $c_0 = (q_0, x_0) \in \mathcal{C}$, *and* $\mathcal{C}' = \mathcal{C} \setminus \{c_0\} \uplus \{(q_1, x_1)\} \uplus \{(q_1', x_1')\}$, *with* $x_0 = x_1 + x_1'$.
if t **is join:** $t = ((q_0, q_1), (q_0', q_1))$, *there exist* $c_0 = (q_0, x_0) \in \mathcal{C}$ *and* $c_0' = (q_0', x_0') \in \mathcal{C}$, *and* $\mathcal{C}' = \mathcal{C} \setminus \{c_0\} \setminus \{c_0'\} \uplus \{(q_1, x_0 + x_0')\}$.

Definition 4. $\mathbb{Z}$ *and* $\mathbb{N}$*-Reachability Problem.*
Let S *be a VASS-SJ,* $\mathcal{C}_0$, $\mathcal{C}_1$ *be two configurations on* S. *The* $\mathbb{Z}$-reachability problem *(respectively* $\mathbb{N}$-reachability problem*) for* S, $\mathcal{C}_0$, $\mathcal{C}_1$ *is the following:*
Does there exist a finite sequence of transitions $t_0, \cdots, t_k$ *of* S *such that* $(S, \mathcal{C}_0) \rightarrow_{t_0} \cdots \rightarrow_{t_k} (S, \mathcal{C}_1)$*? (resp. such that* $(S, \mathcal{C}_0) \rightarrow_{t_0} \cdots \rightarrow_{t_k} (S, \mathcal{C}_1)$, *with only positive configurations?)*

Theorem 1. *The* $\mathbb{Z}$*-reachability problem for VASS-SJ is decidable.*

1.3 Generalized Configurations and Transition Trees

We extend the three types of transitions of a VASS-SJ with a new idle (id) transition type. We use this new transition type as a tool to sequentialize the execution semantics of VASS-SJ in a graph theoretic way.

Definition 5. *Generalized configurations.*
For $m \geq 1$, *we consider the usual product order* $\leq$ *on* $(\mathbb{N} \cup \infty)^m$. *Given a VASS-SJ* $S = (G, T, m, v)$, *with* $G = (Q, A)$, *a* generalized single configuration *of* S *is a 2-tuple* $g = (q, x)$, *where* $q \in Q$ *is a* state *and* $x \in (\mathbb{N} \cup \infty)^m$ *is a* generalized value. *For two generalized single configurations* g *and* g', *we write* $g \leq g'$ *if and only if* $g = (q, x)$, $g' = (q, x')$ *and* $x \leq x'$. *A* generalized configuration *is a 2-tuple* $(\mathcal{G}, M_{\mathcal{G}})$, *where* $\mathcal{G}$ *is a finite set of generalized single configurations of* S, *and* $M_{\mathcal{G}} : \mathcal{G} \rightarrow \mathbb{N} \cup \infty$ *is a multiplicity function. In other words, a generalized configuration is a finite multiset whose elements may have finite or infinite multiplicity. Abusing notations, , we will often write a generalized configuration with a set notation* $\mathcal{G} = \{c_1^{k_1}, \cdots, c_n^{k_n}\}$, *where the* k_i *are the multiplicities. The firing of a transition* t *of* S *is naturally extended over generalized configurations: Let* $\mathcal{G}$ *and* $\mathcal{G}'$ *be two generalized configurations of* S. *The* firing *of* t *in* $(S, \mathcal{G})$ *is the relation* $(S, \mathcal{G}) \rightarrow_t (S, \mathcal{G}')$, *where*

if t **is regular:** $t = (q_0, q_1)$, *there exists* $c = (q_0, x_0) \in \mathcal{G}$, *and* $\mathcal{G}' = \mathcal{G} \setminus \{c\} \uplus \{(q_1, x_0 + v(t))\}$.
if t **is split:** $t = ((q_0, q_1), (q_0, q_1'))$, *there exists* $c_0 = (q_0, x_0) \in \mathcal{G}$, *and* $\mathcal{G}' = \mathcal{G} \setminus \{c\} \uplus \{(q_1, x_1)\} \uplus \{(q_1', x_1')\}$, *with* $x_0 = x_1 + x_1'$ *if* $x_0 \in \mathbb{N}$, $x_0 = x_1 = x_2 = \infty$ *otherwise.*
if t **is join:** $t = ((q_0, q_1), (q_0', q_1))$, *there exist* $c_0 = (q_0, x_0) \in \mathcal{G}$ *and* $c_0' = (q_0', x_0') \in \mathcal{G}$, *and* $\mathcal{G}' = \mathcal{G} \setminus \{c_0\} \setminus \{c_0'\} \uplus \{(q_1, x_0 + x_0')\}$.
if t **is** (id)**:** $\mathcal{G} = \mathcal{G}'$.

Definition 6. *Base of a generalized configuration*
Let $S = (G, T, m, v)$ *be a VASS-SJ, and* $\mathcal{G}$ *be a generalized configuration of* S. *A* Base *of* $\mathcal{G}$ *is a finite set* $B_\mathcal{G}$ *of vertices, each vertex being labelled with some* g^k, *where* $g \in \mathcal{G}$ *and* $k \in \{1, \infty\}$, *such that:*

- *for every generalized single configuration* $g \in \mathcal{G}$, $M_\mathcal{G}(g)$ *is the sum of the multiplicities of labels of vertices in* $B_\mathcal{G}$ *labelled with* g^k, *for any* k,
- *for every vertex* v *in* $B_\mathcal{G}$ *labelled with* g^k, *the corresponding* g *is in* $\mathcal{G}$, *and*
- *for every generalized single configuration* $g \in \mathcal{G}$ *with* ∞ *multiplicity, exactly one vertex* v *in* $B_\mathcal{G}$ *is labelled with* g^∞.

$B_\mathcal{G}$ *is in* normal form *if, for any* $g \in \mathcal{G}$ *with* ∞ *multiplicity, no vertex is labelled with* g^1. *The firing of a transition* t *of* S *over generalized configurations is extended over their bases: if* $(S, \mathcal{G}) \to_t (S, \mathcal{G}')$, *and* B *and* B' *are bases of* $\mathcal{G}$ *and* $\mathcal{G}'$ *respectively, we define* $(S, B) \to_t (S, B')$ *as a bipartite graph* $(B \uplus B', E)$, *where,*

- *all edges but one (if* t *is regular) or two (if* t *is join or split) are* (id) *edges, from* $v \in B$ *labelled with* g^k *to* $w \in B'$ *labelled with the same* g^k,
- *all vertices of* B *labelled with* g^1 *are source of exactly one edge, all vertices of* B' *labelled with* g^1 *are target of exactly one edge,*
- *all vertices of* B *labelled with* g^∞ *are source of an* (id) *edge, all vertices of* B' *labelled with* g^∞ *are target of an* (id) *edge,*
- *if* t *is regular, there exists one edge* (v, w), *where* $v \in G$ *is labelled with* g^k, $w \in G'$ *is labelled with* h^l, *and* $(S, \{g^1\}) \to_t (S, \{h^1\})$,
- *if* t *is join, there exist two edges* (v_1, w) *and* (v_2, w), *where* $v_1 \in G$ *is labelled with* ${g_1}^{k_1}$, $v_2 \in G$ *is labelled with* ${g_2}^{k_2}$, $w \in G'$ *is labelled with* h^l *and* $(S, \{{g_1}^1\} \uplus \{{g_2}^1\}) \to_t (S, \{h^1\})$, *and,*
- *if* t *is split, there exist two edges* (v, w_1) *and* (v, w_2), *where* $v \in G$ *is labelled with* g^k, $w \in G'$ *is labelled with* ${h_1}^{l_1}$, $w' \in G'$ *is labelled with* ${h_2}^{l_2}$ *and* $(S, \{g^1\}) \to_t (S, \{{h_1}^1\} \uplus \{{h_2}^1\})$.

Definition 7. $\preceq$ *order relation, welding graphs.*
Let $S = (G, T, m, v)$ *be a VASS-SJ,* $\mathcal{G}$ *and* $\mathcal{G}'$ *be two generalized configurations of* S *with bases* $B_\mathcal{G}$ *and* $B_{\mathcal{G}'}$. *Then,* $B_\mathcal{G} \preceq B_{\mathcal{G}'}$ *if and only if there exists a directed graph* $G_{B_\mathcal{G} \preceq B_{\mathcal{G}'}} = (V, E)$, *called* welding graph, *such that:*

- $V = B_\mathcal{G} \uplus B_{\mathcal{G}'}$,
- $E \subseteq B_\mathcal{G} \times B_{\mathcal{G}'}$,

- $(v, v') \in E$ *only if* $v \in B_{\mathcal{G}}$ *is labelled with* g^k, $v' \in B_{\mathcal{G}'}$ *is labelled with* $g'^{k'}$, *and* $g \leq g'$,
- *for all* $v \in B_{\mathcal{G}}$ *labelled with* g^∞, *there exists at least one* $v' \in B_{\mathcal{G}'}$ *with* $(v, v') \in E$ *labelled with* g'^∞, *and*
- *for all* $v \in B_{\mathcal{G}}$ *labelled with* g^1, *there exists exactly one* $v' \in B_{\mathcal{G}'}$ *with* $(v, v') \in E$, *and* v' *is labelled with* g'^1.

Note that a given relation $B_{\mathcal{G}} \preceq B_{\mathcal{G}'}$ *may have different welding graphs. If, moreover, for any edge* $(v_1, v_0) \in G_{B_{\mathcal{G}} \preceq B_{\mathcal{G}'}}$, *with* v_1 *labelled with* $g_1^{k_1}$ *and* v_0 *labelled with* $g_0^{k_0}$, *each coordinate in* $(g_0 - g_1)$ *is either* 0 *or* ∞, *we write* $g_1 \leq_\infty g_0$ *and* $B_{\mathcal{G}} \preceq_\infty B_{\mathcal{G}'}$ *respectively.*

Definition 8. *Quotient graph, quotient tree.*
Let $G = (V, E)$ *be a finite directed graph. A partitioning of* G *is a division of its vertices into disjoint subsets* $\mathcal{B} = \{B_1, \cdots, B_k\}$. *The* quotient graph *induced by the partitioning, written,* $G/\mathcal{B}$, *is the graph* $G' = (B, E')$ *where* $(B_i, B_j) \in E'$ *if and only if there exists* $v_i \in B_i$ *and* $v_j \in B_j$ *such that* $(v_i, v_j) \in E$. *If* $G/\mathcal{B}$ *is a tree we call it the* quotient tree *of* G *induced by* $\mathcal{B}$.

Definition 9. *pairing relation.*
Let $G = (V, E)$ *be a directed graph. A* pairing relation *on* E *is an symmetric relation on* E *such that:*

- *only one edge can be paired with any given edge, and*
- (v_1, w_1) *and* (v_2, w_2) *are paired only if either* $v_1 = v_2$, *or* $w_1 = w_2$.

Definition 10. *Generalized Transition Tree (GTT)*
Let $S = (G, T, m, v)$, *with* $G = (Q, A)$ *be a VASS-SJ. A* Generalized transition tree *(GTT)* $(G, \mathcal{B})$ *on* S *is a labelled acyclic directed finite graph* $G = (V, E)$, *together with a pairing relation on* E, *and a partitioning* $\mathcal{B} = \{B_1, \cdots, B_k)$ *of* V *such that:*

- *each vertex of* V *is labelled with a generalized single configuration* g *on* S, *together with a multiplicity* 1 *or* ∞,
- *each edge of* E *is labelled with a transition* t *of* S, *or with* (id),
- $G/\mathcal{B}$ *is a quotient tree, and*
- *for any edge* (B_i, B_j) *of* $G/\mathcal{B}$,
  1. *all edges from* B_i *to* B_j *but at most one (labelled with a regular* t*) or two (labelled with a join or split* t*) are labelled with* (id), *and*
  2. *there exists a base* B'_j *such that* $B_i \rightarrow_t B'_j$, *and* B_j *can be obtained from* B'_j *by replacing finite coordinates or finite multiplicities with* ∞ *ones.*

Moreover, $(G, \mathcal{B})$ *is in* normal form *if all bases in* $\mathcal{B}$ *are in normal form. The* normal form *of* $(G, \mathcal{B})$ *is the GTT obtained from* $(G, \mathcal{B})$ *by merging together all vertices of the same base* B_i *in* B *labelled with* g^∞ *or* g^1 *for the same* g, *as long as there exists one vertex labelled with* g^∞. *When* $G/\mathcal{B}$ *is a sequence,* G *is refereed as a* Generalized transition sequence *(GTS). When moreover all bases of* $\mathcal{B}$ *are bases of (positive) configurations,* G *is refereed as a* (positive) transition sequence.

Remark 1. Without loss of generality, the $\mathbb{Z}$ (respectively $\mathbb{N}$) Reachability Problem can be formulated as follows: Let S be a VASS-SJ, $\mathcal{C}_0$, $\mathcal{C}_1$ be two configurations on S of base B_0 and B_1. Does there exists a transition sequence (resp. positive transition sequence) $(G, \mathcal{B})$ from B_0 to B_1 in S?

2 Karp and Miller Tree

2.1 Definitions

We present in this section some natural extensions of classical notions used for PNs and VASS to our setting of VASS-SJ. We do not only introduce ∞ values in single configurations, as in the classical case, but also ∞ multiplicities of single configurations, to denote the fact that, from some point, a value or a multiplicity can be raised to an arbitrary high level. We present now the lifting operation, which allows us to replace a finite value or multiplicity by an ∞ one when applicable.

Definition 11. *Single Liftings.*
Let S be a VASS-SJ. Let $(G, \mathcal{B})$ be a GTT on S. Let $B_i, B_n \in B$, where B_n is a leaf of the quotient tree $G/\mathcal{B}$. We write $B_i \ll B_n$ if and only if $B_i \preceq B_n$ and there exists a path from B_i to B_n in $G/\mathcal{B}$. Similarly we write $B_i \ll_\infty B_n$ if and only if $B_i \preceq_\infty B_n$ and there exists a path from B_i to B_n in $G/\mathcal{B}$. Assume $B_i \ll B_n$ and let $G_{B_i \preceq B_n} = (B_i \uplus B_n, E)$ be the corresponding welding graph. The single lifting *of (B_n, B_i) in $(G, \mathcal{B})$, denoted as $\#(B_n, B_i)^{G_{B_i \preceq B_n}}$ is the following operation.*

- *$\forall (v, v') \in E$, v labelled with $(q, x)^k$, v' labelled with $(q, x')^k$, $\forall 1 \leq j \leq m$, if $x_j < x'_j$, replace x'_j with ∞,*
- *for any point $v \in B_n$ isolated in $G_{B_i \preceq B_n}$, labelled with g^1, make v' a copy of v, and label v' with g^∞.*

Note that different choices choices for the welding graph $G_{B_i \preceq B_n}$ yield actually different results: a point isolated in one welding graph may not be isolated in another one. In such a case, making a copy of the isolated point, and modifying the label of the copy only ensures that two such single liftings commute. Note also that the number of possible choices is clearly exponentially bounded in the size of the bases.

Definition 12. *Liftings.*
Let S be a VASS-SJ. Let $(G, \mathcal{B})$ be a GTT on S. Let $B_n \in B$ be a leaf of the quotient tree $G/\mathcal{B}$. Let $B'_1, \cdots, B'_k$ be the finite (possibly empty) sequence of bases in $\mathcal{B}$, in order increasing from the root of $(G/\mathcal{B})$ to B_n, such that, for any B'_i in the sequence, $B'_i \ll B_n$ in $(G, \mathcal{B})$. If the sequence $B'_1, \cdots, B'_k$ is not empty, the lifting transition sequence of B_n is the GTS $= (G', E)$, with $E = \{E_0, \cdots, E_i\}$, inductively defined as follows

1. *$B_n = E_0$ is the (finite) set of vertices of G' of in-degree 0, with the same labels as in $(G, \mathcal{B})$.*

2. *For $i = 1, \cdots, k$, let $E_{i-1} \in E$ be the leaf of G'/E, of depth $i-1$. Extend (G', E') with a GTS from E_{i-1} to a copy E_i of E_{i-1}, (with only (id) edges). Then, For all welding graphs $G_{B'_i \preceq E_i}$ corresponding to $B'_i \preceq E_i$, perform the single lifting $\#(E'_i, B'_j)^{G_{B'_i \preceq E_i}}$.*

If the sequence $B'_1, \cdots, B'_k$ is empty, the lifting transition sequence of B_n is restricted to B_n. Now, the lifting of B_n in $(G, \mathcal{B})$, denoted as $\#(B_n)$, is the result of extending $(G, \mathcal{B})$ with the lifting transition sequence of B_n, and normalizing.

Now, we introduce our construction of a Karp and Miller Tree.

Definition 13. *Karp and Miller Tree.*
Let S be a VASS-SJ and B be a base of a configuration $\mathcal{C}$ of S. The Karp and Miller tree *$\mathfrak{T} = (V, E, \mathcal{B})$ on (S, B) is a GTT $(G, \mathcal{B})$, where $G = (V, E)$, constructed inductively as follows.*

1. *$B_1 \in \mathcal{B}$ is B.*
2. *Let $B_j \in \mathcal{B}$ base of $\mathcal{G}_j$ in G. If there exists $B_i \in \mathcal{B}$ base of the same $\mathcal{G}_j$ in G, and a path from B_i to B_j in $G/\mathcal{B}$, the vertices of B_j have out-degree 0 in G (hence B_j has out-degree 0 in $G/\mathcal{B}$). Otherwise,*
3. *assume there exists a transition t and a generalized configuration $\mathcal{G}_k$ with base B_k in normal form, such that $\mathcal{B}_j \rightarrow_t \mathcal{B}_k$. Then, extend $\mathfrak{T}$ with the GTS $(B_j \rightarrow_t B_k, B_j \uplus B_k)$, and perform the lifting $\#(B_k)$ in $\mathfrak{T}$.*

This inductive construction halts when no new base can be added with these rules.

The definition above induces the following property: $\mathcal{B}_i \ll \mathcal{B}_n \Rightarrow \mathcal{B}_i \ll_\infty \mathcal{B}_n$.

2.2 Constructibility of the Karp and Miller Tree

Lemma 1. *The following statements are true:*

1. *Let $u_n \in (\mathbb{N} \uplus \infty)^m$, $n \in \mathbb{N}$, be an infinite sequence of m-tuples for some $m \in \mathbb{N}$. Then, there exists an infinite sub-sequence u'_n, $n \in \mathbb{N}$ of u_n that is increasing for the order relation $\leq$.*
2. *Let S be a VASS-SJ. Let $(G, \mathcal{B})$ be an infinite GTS on S, with $\mathcal{B} = (B_n)$, $n \in \mathbb{N}$. Then, there exists an infinite sub-sequence (B'_n), $n \in \mathbb{N}$ of (B_n) that is increasing for the order relation $\preceq$ in $(G, \mathcal{B})$.*

Proof. The proof is based on well quasi-orders and Higman's Lemma [Hig52].

Theorem 2. *Let S be a VASS-SJ and B be a base of a configuration $\mathcal{C}$ of S. The Karp and Miller tree $\mathfrak{T}$ on (S, B) is finite, and can be effectively constructed.*

Proof. The proof is based on Lemma 1 and Koenig's Lemma.

3 Towards $\mathbb{N}$-reachability

3.1 Relating Transition Sequences and GTS

Definition 14. *Expansion and Composition of GTSs*
Let S be a VASS-SJ. Let $(G_2, \mathcal{B}_2)$ with $\mathcal{B}_2 = B_2^1, \cdots, B_2^g$ be a GTS with initial base B_2^1 and final base B_2^g. Let B_1^f be a base of a generalized configuration of S, such that $B_2^1 \preceq B_1^f$ with welding graph $G_{B_2^1 \preceq B_1^f}$. An expansion *of $(G_2, \mathcal{B}_2)$ with respect to the welding graph $G_{B_2^1 \preceq_\infty B_1^f}$, is a GTS $(H, \mathcal{D})$, with $\mathcal{D} = D_1, \cdots, D_f$, where, for all $i = 1, \cdots, f$, $B_2^i \preceq D_i$ with welding graph $G_{B_2^i \preceq D_i}$, such that, for all $i = 1, \cdots, f-1$:*

- *for any vertex $w_i \in D_i$ isolated in the welding graph $G_{B_2^i \preceq D_i}$, there exists a vertex $w_{i+1} \in D_{i+1}$ isolated in the welding graph $G_{B_2^{i+1} \preceq D_{i+1}}$ and an (id) edge (w_i, w_{i+1}) in H.*
- *for any vertices $v_i \in B_2^i$, $v_{i+1} \in B_2^{i+1}$, such that there exists an edge (v_i, v_{i+1}) labelled with a transition t (respectively labelled with (id), j, s) in G, there exist two vertices $w_i \in D_i$ and $w_{i+1} \in D_{i+1}$, two edges $(v_i, w_i) \in G_{B_2^i \preceq D_i}$ and $(v_{i+1}, w_{i+1}) \in G_{B_2^{i+1} \preceq D_{i+1}}$, and an edge (w_i, w_{i+1}) labelled with the same transition t (respectively labelled with (id), j, s) in H.*

When, $B_2^1 \preceq_\infty B_1^f$, this expansion is unique and, for all $i = 1, \cdots, f$, $B_2^i \preceq_\infty D_i$. Let now $(G_1, \mathcal{B}_1)$, with $\mathcal{B}_1 = B_1^1, \cdots, B_1^f$ be a GTS. The composition *of $(G_1, \mathcal{B}_1)$ and $(G_2, \mathcal{B}_2)$ with respect to the welding graph $G_{B_2^1 \preceq B_1^f}$, denoted as $(G_1, \mathcal{B}_1).G_{B_2^1 \preceq B_1^f}(G_2, \mathcal{B}_2)$, or simply as $(G_1, \mathcal{B}_1).(G_2, \mathcal{B}_2)$ when the welding graph $G_{B_2^1 \preceq B_1^f}$ is clear from context, is obtained by extending $(G_1, \mathcal{B}_1)$ with the normal form of an expansion of $(G_2, \mathcal{B}_2)$ with respect to the welding graph $G_{B_2^1 \preceq B_1^f}$. When $B_2^1 \preceq_\infty B_1^f$, this composition is uniquely defined.*

Definition 15. *Path in the Karp and Miller Tree*
Let S be a VASS-SJ and $\mathcal{C}$ be a configuration of S with base B. Let $\mathfrak{T} = (V, E, \mathcal{B})$ be the Karp and Miller tree on (S, B). A path *P in $\mathfrak{T}$ is a sequence $(B_1, G_1), \cdots, (B_k, G_k)$ of bases and graphs, such that for all $i = 1, \cdots, k-1$, one of the following holds:*

1. *(B_i, B_{i+1}) is an edge in $\mathfrak{T}/\mathcal{B}$, and G_i is the empty graph, or*
2. *there exists $B_i' \ll_\infty B_i$ in $\mathfrak{T}$, such that (B_i', B_{i+1}) is an edge in $\mathfrak{T}/\mathcal{B}$, and G_i is a welding graph $G_{B_i' \ll_\infty B_i}$. If moreover $B_i' \neq B_i$, we also require that B_i is below B_{i+1} in $\mathfrak{T}/\mathcal{B}$ (i.e. in the same branch of the tree).*

When only (1) holds, for all $i = 1, \cdots, k-1$, P is said to be simple*. From the path P, we also define inductively the following GTSs:*

- *$(H, \mathcal{D})_1$ is the restriction of $\mathfrak{T}$ to $B_1 \uplus B_2$, and*
- *for $i = 1, \cdots, k-1$, $(H, \mathcal{D})_{i+1}$ is the composition of $(H, \mathcal{D})_i$ with the restriction of $\mathfrak{T}$ to $B_i \uplus B_{i+1}$ when (B_i, B_{i+1}) is an edge in $\mathfrak{T}/\mathcal{B}$, or the composition of $(H, \mathcal{D})_i$ with the restriction of $\mathfrak{T}$ to $B_i' \uplus B_{i+1}$ with respect to the welding graph G_i when (B_i', B_{i+1}) is an edge in $\mathfrak{T}/\mathcal{B}$ and $B_i' \ll B_i$ in $\mathfrak{T}$.*

Then, $(H, \mathcal{D})_k$ is uniquely defined, and is the GTS associated to P in $\mathfrak{T}$.

Definition 16. *Transition sequence compatible with a GTS.*
Let S be a VASS-SJ. Let $(G, \mathcal{B})$, with $\mathcal{B} = B_1, \cdots, B_f$, be a GTS on S, with initial base B_1 and final base B_f. Let $\mathcal{C}_1$ and $\mathcal{C}_f$ be two configurations of S with bases D_1 and D_f, and $(P, \mathcal{D})$ be a transition sequence on S, with initial base D_1 and final base D_f, where $\mathcal{D} = D_1, \cdots, D_f$. $D_1, \cdots, D_f$ are compatible *with $B_1, \cdots, B_f$ if and only if there exist functions $f_{D_i \to B_i} : D_i \to B_i$, for $i = 1, \cdots, f$, called* compatibility *functions, such that, for all $i = 1, \cdots, f$*

1. *$\forall v \in V_i$ labelled with $(q, x_1, \cdots, x_m)$, $f_{D_i \to B_i}(v)$ is labelled with h^l with $h = (q, y_1, \cdots, y_m)$ and, for $j = 1, \cdots, m$:*
   - *$y_j = \infty \Rightarrow x_j \geq 1$, and*
   - *$y_j \neq \infty \Rightarrow x_j = y_j$, and*
2. *for all $w \in B_i$ labelled with h^1, $|f^{-1}_{D_i \to B_i}(w)| = 1$,*
3. *for all $w \in B_i$ labelled with h^∞, $|f^{-1}_{D_i \to B_i}(w)| \geq 1$,*

$(P, \mathcal{D})$ is compatible *with $(G, \mathcal{B})$ if and only if $D_1, \cdots, D_f$ are compatible with $B_1, \cdots, B_f$, and the normal form of $(G, \mathcal{B})$ can be obtained from $(P, \mathcal{D})$ by replacing some finite coordinates or multiplicities in P with infinite ones, and normalizing.*

3.2 Positive Transition Sequences and the Karp and Miller Tree

The two results below show how the Karp and Miller tree captures effectively in a synthetic way all $\mathbb{N}$-reachable positive configurations, bounded or not-bounded. Unbounded configurations appear in the construction if (Proposition 1) and only if (Theorem 3) a path to this configuration in the Karp and Miller tree, and positive transition sequence compatible with this path, exist.

Proposition 1. *Let S be a VASS-SJ and $\mathcal{C}_1$ be a configuration of S with base B_1. Let $\mathfrak{T} = (V, E, \mathcal{B})$ be the Karp and Miller tree on (S, B_1). Let $(H, \mathcal{D})$ be a positive transition sequence of S, where $\mathcal{D} = D_1, \cdots, D_k$ with $D_1 = B_1$. Then, there exists a path P in $\mathfrak{T}$ of length k, such that $(H, \mathcal{D})$ is compatible with the GTS associated to P.*

Proof. The proof is by induction on k, and by construction of $\mathfrak{T}$.

Theorem 3. *Let S be a VASS-SJ and $\mathcal{C}$ be a configuration of S, of base B_1. Let $\mathfrak{T} = (V, E, \mathcal{B})$ be the Karp and Miller tree on (S, B_1). Let $B_i \in \mathcal{B}$. Then, for any $N \in \mathbb{N}$, there exist a path P^N from B_1 to B_i in $\mathfrak{T}$, a positive transition sequence $(P_i^N, \mathcal{D}_i^N)$ of S with initial base $D_1 = B_1$ and final base D_l, compatible with the GTS associated to P^N, and a compatibility function $f_{D_l \to B_i} : D_l \to B_i$ such that:*

1. *$\forall v \in D_l$ labelled with $(q, x_1, \cdots, x_m)$, $f_{D_l \to B_i}(v)$ is labelled with h^t with $h = (q, y_1, \cdots, y_m)$ and, for $i = 1, \cdots, m$:*
   - *$y_i = \infty \Rightarrow x_i \geq N$, and*

– $y_i \neq \infty \Rightarrow x_i = y_i$, *and*

2. *for all* $w \in B_i$ *labelled with* h^1, $|f^{-1}_{D_l \to B_i}(w)| = 1$,
3. *for all* $w \in B_i$ *labelled with* h^∞, $|f^{-1}_{D_l \to B_i}(w)| \geq N$.

Proof. The proof is by induction on the depth of B_i in $\mathfrak{T}/\mathcal{B}$.

Now, in order to obtain a $\mathbb{N}$-reachability algorithm for a VASS-SJ S from a base B_i to a base B_f, the idea would be, following [Reu88], to obtain two positive transition sequences, from B_i to a base D_l in S, and symmetrically from B_f to a base D'_m in S^{-1}, with values and multiplicities large enough so that one may be able to transform a transition sequence from B_f to D_l in S into a positive one. The composition of three such sequences would then provide a positive transition sequence from B_i to B_f in S.

References

[dGGS04] Philippe de Groote, Bruno Guillaume, and Sylvain Salvati. Vector addition tree automata. In *LICS*, pages 64–73. IEEE Computer Society, 2004.

[DJLL09] Stéphane Demri, Marcin Jurdzinski, Oded Lachish, and Ranko Lazic. The covering and boundedness problems for branching vector addition systems. In Ravi Kannan and K. Narayan Kumar, editors, *FSTTCS*, volume 4 of *LIPIcs*, pages 181–192. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2009.

[dNM10] Paulin Jacobé de Naurois and Virgile Mogbil. Rewriting systems for reachability in vector addition systems with pairs. In Antonín Kucera and Igor Potapov, editors, *RP*, volume 6227 of *Lecture Notes in Computer Science*, pages 133–145. Springer, 2010.

[FG12] Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, part II: Complete WSTS. *Logical Methods in Computer Science*, 8(3:28), September 2012.

[Hig52] Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, (3), 2(7):326–336, september 1952.

[Kos82] S. Rao Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In *STOC*, pages 267–281. ACM, 1982.

[Lam92] Jean-Luc Lambert. A structure to decide reachability in petri nets. *Theoretical Computer Science*, 99:79–104, 1992.

[Laz10] Ranko Lazic. The reachability problem for branching vector addition systems requires doubly-exponential space. *Inf. Process. Lett.*, 110(17):740–745, 2010.

[May81] Ernst W. Mayr. An algorithm for the general petri net reachability problem. In *STOC*, pages 238–246. ACM, 1981.

[May84] Ernst W. Mayr. An algorithm for the general petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.

[Reu88] Christophe Reutenauer. *The Mathematics of Petri Nets*. Masson, 1988.

[VGL05] Kumar Neeraj Verma and Jean Goubault-Larrecq. Karp-miller trees for a branching extension of vass. *Discrete Mathematics & Theoretical Computer Science*, 7(1):217–230, 2005.

Lowness for uniform Kurtz randomness

Takayuki Kihara[1] and Kenshi Miyabe[2]

[1] Japan Advanced Institute of Science and Technology, Japan
kihara.takayuki.logic@gmail.com
[2] Research Institute for Mathematical Sciences, Kyoto University, Japan
kmiyabe@kurims.kyoto-u.ac.jp

Abstract. We propose studying uniform Kurtz randomness, which is the uniform relativization of Kurtz randomness. One advantage of this notion is that lowness for uniform Kurtz randomness has many characterizations, such as those via complexity, martingales, Kurtz tt-traceability, and Kurtz dimensional measure.

Keywords: algorithmic randomness, lowness for randomness, effective dimension

1 Introduction

One of the major topics in algorithmic randomness is "lowness". For a given randomness notion R, A is said to be low for R if every R-random set is R-random relative to A, that is, A does not have enough computational power to derandomize a random set. For instance, lowness for ML-randomness has many characterization such as K-triviality, lowness for K and being a base for ML-randomness [16, 11].

Lowness for Schnorr randomness has previously been studied in the literature. Some studies, however, have suggested that uniform Schnorr randomness (the uniform relativization of Schnorr randomness) is the proper relativization because it satisfies van Lambalgen's theorem [14, 15] and has natural lowness properties [9, 14, 13]. Similar phenomena have been found for other notions of randomness [1, 2, 5]. In particular, the second author and Rute [15] have claimed that uniform relativization is the correct relativization for all randomness notions.

In this paper we study a version of Kurtz randomness. There are already some known results on lowness for Kurtz randomness [8, 18, 10]. Again, however, van Lambalgen's theorem does not hold for Kurtz randomness [9], and it seems that uniform Kurtz randomness (the uniform relativization of Kurtz randomness) is a more natural notion. (We will show in another paper that van Lambalgen's theorem holds for uniform Kurtz randomness in a weaker sense, hence Kurtz randomness and uniform Kurtz randomness are different.) In this paper, we show that lowness for uniform Kurtz randomness has many characterizations, which advocates for its naturalness.

The overview of this paper is as follows. In Section 3 we introduce uniform Kurtz randomness defined by tests and characterize it via complexity and martingales. In Section 4 we introduce the notion of Kurtz h-dimensional measure zero where h is an order, and give characterizations via complexity and martingales. In Section 5 we characterize lowness for uniform Kurtz randomness via Kurtz h-dimensional measure zero and Kurtz tt-traceability. To prove this, we make use of the svelte tree introduced in Greenberg-Miller [10].

2 Preliminaries

We say that $n > 0$ is the index of a finite set $\{x_1, \cdots, x_r\}$ of natural numbers if $n = 2^{x_1} + 2^{x_2} + \cdots + 2^{x_r}$, while 0 is the index of $\emptyset$. In the following we often identify a finite set with its index. We also often identify $\sigma \in 2^{<\omega}$ with the natural number n represented by 1σ in binary representation. An *order* is a nondecreasing unbounded function from ω to ω. We denote the empty string by ϵ.

We recall some results on Kurtz randomness. The reader may refer to [7, 17] for details. Let μ be the uniform measure on the Cantor space 2^ω. A set $A \in 2^\omega$ is *weakly 1-random*, or *Kurtz random*, if it is contained in every c.e. open set with measure 1 [12]. A *Kurtz null test* is a sequence $\{[\![f(n)]\!]\}$ such that $f : \omega \to (2^{<\omega})^{<\omega}$ is a computable function and $\mu([\![f(n)]\!]) \leq 2^{-n}$. A set is Kurtz random if and only if it passes all Kurtz null tests [19]. A *computable measure machine* is a prefix-free machine M such that $\mu([\![\mathrm{dom}M]\!])$ is computable [6].

3 Uniform Kurtz randomness

The definition of uniform relativization [15] requires some definitions in computable analysis [20, 4, 3, 21]. Let τ be the class of open sets on 2^ω. A partial function $f :\subseteq 2^\omega \to \tau$ is *computable* if there is a partial computable function $\psi :\subseteq 2^\omega \times \omega \to (2^{<\omega})^{<\omega}$ such that $f(Z) = \bigcup_{n\in\omega} [\![\psi(Z, n)]\!]$ for each Z. If such a function ψ is total, then f is also called *total*.

Definition 1. *A* uniform Kurtz test *is a total computable function $f : 2^\omega \to \tau$ such that $\mu(f(Z)) = 1$ for all $Z \in 2^\omega$. A set $B \in 2^\omega$ is* Kurtz random uniformly relative to *$A \in 2^\omega$ if $B \in f(A)$ for each uniform Kurtz test f.*

Definition 2. *A* uniform Kurtz null test *is a computable function $f : 2^\omega \times \omega \to (2^{<\omega})^{<\omega}$ such that, for each $Z \in 2^\omega$ and $n \in \omega$, $\mu([\![f(Z, n)]\!]) \leq 2^{-n}$. For a fixed set X, we also say that $\{[\![f(X, n)]\!]\}_{n\in\omega}$ is* a Kurtz null test uniformly relative to *X.*

We give characterizations of uniform Kurtz randomness via machines and martingales. Recall the following characterization of Kurtz randomness: A set X is not Kurtz random if and only if there is a computable measure machine M and a computable function f such that, for all n, $K_M(X \upharpoonright f(n)) < f(n) - n$ [8] if and only if there are a computable martingale d and a computable order h such that $d(X \upharpoonright n) > h(n)$ for all n [19, 8].

Proposition 1. *The following are equivalent for sets A and B.*

(i) *A is not Kurtz random uniformly relative to B.*
(ii) *There are an oracle prefix-free machine M and a computable function h such that $Z \mapsto \mu(\mathrm{dom}M^Z)$ is a computable function and $K_{M^B}(A \upharpoonright h(n)) < h(n) - n$ for all $n \in \omega$.*
(iii) *There are a $\mathbb{Q}$-valued martingale $d \leq_{tt} B$ and a computable order h such that $d(A \upharpoonright n) > h(n)$ for all $n \in \omega$.*

Proof. The proof of (i) $\Longleftrightarrow$ (ii) is a straightforward modification of the unrelativized version in [8].

(i)$\Rightarrow$(iii): Suppose that A is not Kurtz random uniformly relative to B. Then there is a uniform Kurtz null test f such that $A \in \bigcap_n [\![f(B,n)]\!]$. Since f is a total computable function, we can assume the existence of a strictly increasing computable function g such that $g(0) = 0$ and $\sigma \in f(Z,n) \Rightarrow |\sigma| = g(n)$. Let k be a computable order such that $k(0) = 0$ and $k(n) \geq g(k(n-1)) + 1$ for all $n \geq 1$.

We construct a $\mathbb{Q}$-valued martingale d^Z as follows. Let $d^Z(\epsilon) = 2$. At stage $n \geq 1$, we define $d(\sigma)$ for $\sigma \in 2^{<\omega}$ such that $g(k(n-1)) < |\sigma| \leq g(k(n))$. Note that $g(k(0)) = 0$. For each $\tau \in f(Z, k(n-1))$, let $a(\tau)$ be the number of strings $\rho \in f(Z,k(n))$ such that $\tau \prec \rho$. For each $\tau \notin f(Z,k(n-1))$, let $a(\tau) = 0$. Note that a is computable from Z. We assume that, for each $\rho \in f(Z,n)$, there is $\tau \in f(Z,n-1)$ such that $\tau \prec \rho$, whence $\sum_{\tau \in f(Z,k(n-1))} a(\tau) = \#f(Z,k(n)) \leq 2^{g(k(n))-k(n)}$ where the last inequality follows from $\mu([\![f(Z,k(n))]\!]) \leq 2^{-k(n)}$. Let $\sigma \in 2^{<\omega}$ be such that $g(k(n-1)) < |\sigma| \leq g(k(n))$. We define a $\mathbb{Q}$-valued martingale $d^Z(\sigma)$ by

$$d^Z(\sigma) = \begin{cases} d^Z(\sigma \upharpoonright g(k(n-1))) & \text{if } a(\sigma \upharpoonright g(k(n-1))) = 0 \\ e^Z(\sigma) & \text{if there is } \tau \in f(Z,k(n)) \text{ such that } \sigma \preceq \tau \\ \frac{d^Z(\sigma \upharpoonright g(k(n-1)))}{2} & \text{otherwise,} \end{cases}$$

where

$$e^Z(\sigma) = d^Z(\sigma \upharpoonright g(k(n-1))) \left(\frac{1}{2} + \frac{2^{|\sigma| - g(k(n-1))}}{2 \cdot a(\sigma \upharpoonright g(k(n)))} \right).$$

Clearly, $d = d^B \leq_{tt} B$.

First we show that $d(A \upharpoonright g(k(n))) > \left(\frac{3}{2}\right)^n$ for all $n \in \omega$. If $n = 0$, $d(\epsilon) = 2 > 1 = \left(\frac{3}{2}\right)^0$. By assuming that $d(A \upharpoonright g(k(n-1))) > \left(\frac{3}{2}\right)^{n-1}$, we have

$$\begin{aligned} d(A \upharpoonright g(k(n))) = e^B(A \upharpoonright g(k(n))) &\geq d(A \upharpoonright g(k(n-1))) \left(\frac{1}{2} + \frac{2^{g(k(n)) - g(k(n-1))}}{2^{g(k(n)) - k(n) + 1}} \right) \\ &\geq \frac{3}{2} d(A \upharpoonright g(k(n-1))) > \left(\frac{3}{2}\right)^n. \end{aligned}$$

We define a computable order h by

$$h(m) = \left\lfloor \frac{1}{2} \cdot \left(\frac{3}{2}\right)^{n-1} \right\rfloor \text{ where } g(k(n-1)) \leq m < g(k(n)),$$

where $\lfloor x \rfloor$ denotes the largest integer not greater than x. If $m = 0$, then $d(A \upharpoonright m) = d(\epsilon) = 2 > 1 = h(0) = h(m)$. If m satisfies $g(k(n-1)) < m \leq g(k(n))$, then

$$d(A \upharpoonright m) \geq \frac{d(A \upharpoonright g(k(n-1)))}{2} > \frac{1}{2} \cdot \left(\frac{3}{2}\right)^{n-1} \geq h(m).$$

(iii)⇒(i): Assume that $d \leq_{tt} B$. Then there is a truth-table functional Ψ such that Ψ^Z is a $\mathbb{Q}$-valued martingale for each $Z \in 2^\omega$ and $d = \Psi^B$. Let f be a computable order such that $h(f(n)) \geq 2^n$ for all $n \in \omega$. Consider the following clopen set: $C_n^Z = \{\sigma \in 2^{f(n)} \ : \ \Psi^Z(\sigma) \geq 2^n\}$. Then $\mu(C_n^Z) \leq 2^{-n}$ for all $n \in \omega$ and $Z \in 2^\omega$. Since $\Psi^B(A \upharpoonright f(n)) = d(A \upharpoonright f(n)) > h(f(n)) \geq 2^n$ for each n, we have $A \in \bigcap_n C_n^B$. □

4 Kurtz Dimensional Measure

The effectivization of concepts from fractal geometry such as Hausdorff dimension is playing a greater role in algorithmic randomness theory (see [7, Section 13]). In this section, we introduce and give some characterizations of the notion of effective Hausdorff-like dimension, which we call Kurtz h-dimensional measure zero.

Definition 3. *For an order $h : \omega \to \omega$, a set $E \subseteq 2^\omega$ is* Kurtz h-dimensional measure zero *if there is a computable sequence $\{C_n\}_{n\in\omega}$ of finite sets of strings such that*

$$E \subseteq [\![C_n]\!] \text{ and } \sum_{\sigma \in C_n} 2^{-h(|\sigma|)} \leq 2^{-n} \text{ for all } n \in \omega.$$

We also say that $A \in 2^\omega$ is Kurtz h-dimensional measure zero *if $\{A\}$ is Kurtz h-dimensional measure zero.*

Theorem 1. *Let h be any computable order. Then the following are equivalent for a set A.*

(i) *A is Kurtz h-dimensional measure zero.*
(ii) *There are a computable martingale d and a computable order g such that*

$$(\forall n \in \omega)(\exists k \in [g(n), g(n+1)))) \quad d(A \upharpoonright k) \geq 2^n \cdot 2^{k-h(k)}.$$

(iii) *There are a computable measure machine M and a computable order g such that*

$$(\forall n \in \omega)(\exists k \in [g(n), g(n+1)))) \quad K_M(A \upharpoonright k) \leq h(k) - n.$$

Proof. (i)⇒(ii): Suppose that A is Kurtz h-dimensional measure zero via a sequence $\{C_n\}_{n\in\omega}$. Find $t(n+1) \geq 2(n+1)+1$ such that all strings contained in $C_{t(n+1)}$ are longer than any strings of $C_{t(n)}$. Here $t(n) \geq 2n+1$ implies that $\sum_{\sigma \in C_{t(n)}} 2^{-h(|\sigma|)} \leq 2^{-2n-1}$. For each σ, let B_σ be a martingale defined by

$$B_\sigma(\tau) = \begin{cases} 2^{|\tau|-h(|\sigma|)} & \text{if } \tau \preceq \sigma \\ 2^{|\sigma|-h(|\sigma|)} & \text{if } \sigma \prec \tau \\ 0 & \text{otherwise.} \end{cases}$$

Then $d = \sum_n \sum_{\sigma \in C_{t(n)}} 2^n B_\sigma$ is a computable martingale with the initial capital

$$\sum_n \sum_{\sigma \in C_{t(n)}} 2^{n-h(|\sigma|)} = \sum_n 2^n \cdot 2^{-2n-1} \leq \sum_n 2^{-n-1} = 1.$$

Define g to be a computable order such that the length of every string in $C_{t(n)}$ is contained in $[g(n), g(n+1))$. Then, for all $n \in \omega$, there is a $k \in [g(n), g(n+1))$ such that $A \upharpoonright k \in C_{t(n)}$, that is,

$$d(A \upharpoonright k) \geq 2^n B_{A\upharpoonright k}(A \upharpoonright k) = 2^n \cdot 2^{k-h(k)}.$$

(ii)⇒(iii): By our assumption, for every $n \in \omega$, there is $k \in [g(n), g(n+1))$ such that $d(A \upharpoonright k) \geq 2^n 2^{k-h(k)}$. Without loss of generality, we may assume that $d(\epsilon) = 1$. Consider the following clopen set:

$$C_n = \{\sigma \in 2^{<\omega} : |\sigma| \in [g(2n), g(2n+1)), \text{ and } d(\sigma) \geq 2^{2n} 2^{|\sigma| - h(|\sigma|)}\}.$$

Let D_n be an antichain generating C_n. Then

$$\sum_{\sigma \in D_n} 2^{n-h(|\sigma|)} \leq \sum_{\sigma \in D_n} 2^{n-h(|\sigma|)} \frac{2^{-2n} d(\sigma)}{2^{|\sigma|-h(|\sigma|)}} = 2^{-n} \sum_{\sigma \in D_n} 2^{-|\sigma|} d(\sigma) \leq 2^{-n}.$$

Here, the last inequality follows from Kolmogorov's inequality (see [7, Theorem 6.3.3] with our assumption $d(\epsilon) = 1$. Thus, by the KC theorem [7, Theorem 3.6.1], we can construct a computable measure machine M such that, for each $n \in \omega$, $K_M(\sigma) \leq h(|\sigma|) - n$ for each $\sigma \in D_n$. In particular, for all $n \in \omega$, there is $k \in [g(2n), g(2n+1))$ such that $K_M(A \upharpoonright k) \leq h(k) - n$.

(iii)⇒(i): Assume that $K_M(A \upharpoonright k) \leq h(k) - n$ for some $k \in [g(n), g(n+1))$. Consider the sequence $\{C_n\}_{n \in \omega}$ of clopen sets defined by

$$C_n = \{\sigma \in 2^{<\omega} : |\sigma| \in [g(n), g(n+1)), \text{ and } K_M(\sigma) \leq h(|\sigma|) - n\}.$$

Then $A \in \bigcap_n C_n$, and

$$\sum_{\sigma \in C_n} 2^{-h(|\sigma|)} \leq 2^{-n} \sum_{\sigma \in C_n} 2^{-K_M(\sigma)} \leq 2^{-n}.$$

Hence, A is Kurtz h-dimensional measure zero. □

5 Lowness for Uniform Kurtz Randomness

In this section, we give characterizations of lowness for uniform Kurtz randomness. A set $A \in 2^\omega$ is said to be *low for uniform Kurtz randomness* if $X \in 2^\omega$ is uniform Kurtz random relative to A whenever X is Kurtz random. A set $A \in 2^\omega$ is said to be *low for uniform Kurtz tests* if $f(A)$ includes a Kurtz test for every uniform Kurtz test f. For a given order p, a *computable trace with bound* p is

a computable sequence $\{D_n\}_{n\in\omega}$ of finite sets of strings such that $\#D_n \leq p(n)$ for each $n \in \omega$. A computable trace $\{D_n\}_{n\in\omega}$ *Kurtz-traces* a function $f : \omega \to \omega$ if there is a strictly increasing computable sequence $\{l_n\}_{n\in\omega}$ of natural numbers such that

$$(\forall k \in \omega)(\exists n \in [l_k, l_{k+1}))\quad f(n) \in D_n.$$

A set $A \in 2^\omega$ is said to be *Kurtz tt-traceable* if there is a computable order p such that, for every $f \leq_{tt} A$, there is a computable trace with bound p that Kurtz-traces f.

Theorem 2. *The following are equivalent for a set A.*

(i) *A is Kurtz h-dimensional measure zero for every computable order h.*
(ii) *A is low for uniform Kurtz tests.*
(iii) *A is low for uniform Kurtz randomness.*
(iv) *A tt-computes no infinite subset of a Kurtz random set.*
(v) *A is Kurtz tt-traceable.*

Proof. (i)⇒(ii): Let $\{C_n^A\}_{n\in\omega}$ be a Kurtz null test uniformly relative to A, that is, there is a truth table functional Ψ such that $[\![\Psi^A(n)]\!] = C_n^A$, and $\mu(C_n^A) \leq 2^{-n}$. Then there is a computable order u such that, for all $Z \in 2^\omega$ and all $n \in \omega$, the value $\Psi^{Z\restriction u(n)}(n)$ is determined. In particular, $[\![\Psi^{A\restriction u(n)}(n)]\!] = C_n^A$. Let h be a computable order fulfilling $2^{-h(u(n))} \geq 1/(n+1)$ for all $n \in \omega$. Assume that A is Kurtz h-dimensional measure zero. By our assumption, we have a computable sequence $\{D_n\}_{n\in\omega}$ of finite sets of strings such that $A \in [\![D_n]\!]$ and $\sum_{\sigma\in D_n} 2^{-h(|\sigma|)} < 1/(n+1)$ for all $n \in \omega$. Thus, each $\sigma \in D_n$ has length greater than $u(n)$, and moreover D_n contains at most k strings of length $\leq u(n+k)$, since, otherwise,

$$\sum_{\sigma\in D_n} 2^{-h(|\sigma|)} \geq (k+1)2^{-h(u(n+k))} \geq \frac{k+1}{n+k+1} \geq \frac{1}{n+1}.$$

Hence, D_n can be viewed as a finite sequence $\{\sigma_k^n\}_{k<|D_n|}$ of strings such that the length of each σ_k^n is greater than or equal to $u(n+k)$. Thus, there is $k < |D_n|$ such that $A\restriction u(n+k) = \sigma_k^n \restriction u(n+k)$. Inductively define a computable order r by $r(0) = 0$ and $r(n+1) = r(n) + |D_{r(n)}|$. Now $\rho(k)$ is defined by $\sigma_{k-r(n)}^{r(n)} \restriction u(k)$ for each $k \in [r(n), r(n+1))$. Then

$$(\forall n \in \omega)(\exists k \in [r(n), r(n+1)))\ A\restriction u(k) = \rho(k).$$

For all $n \in \omega$ and $k \in [r(n), r(n+1))$, define $E_k \subseteq 2^\omega$ by

$$E_k = \begin{cases} [\![\Psi^{\rho(k)}(k)]\!], & \text{if } \mu([\![\Phi^{\rho(k)}(k)]\!]) \leq 2^{-k}, \\ \emptyset, & \text{otherwise.} \end{cases}$$

Note that $|\rho(k)| = u(k)$ implies that $\Psi^{\rho(k)}(k)$ is defined for all $k \in \omega$ by our assumption for u. Therefore, $\{E_k\}_{k\in\omega}$ is a computable sequence of clopen sets,

and we have

$$C_n^A \subseteq \bigcup_{k=r(n)}^{r(n+1)-1} E_k, \text{ and } \mu\left(\bigcup_{k=r(n)}^{r(n+1)-1} E_k\right) \leq 2^{-r(n)+1} \leq 2^{-n+1}.$$

Consequently, for $B_n = \bigcup_{r(n-1)<t\leq r(n)} E_t$, the sequence $\{B_n\}_{n\in\omega}$ is a Kurtz null test such that $\bigcap_n C_n^A \subseteq \bigcap_n B_n$. In other words, A is low for uniform Kurtz null tests.

(ii)⇒(iii): Obvious.

(iii)⇒(iv): Let $I \subseteq \omega^{\leq\omega}$ be the set of (finite or infinite) strings $\sigma \in \omega^{\leq\omega}$ which are strictly increasing, that is, $\sigma(n) < \sigma(n+1)$ for each $n \in \omega$. Let $\mathrm{rng}(\sigma)$ denote the range of $\sigma \in I$, so that $\mathrm{rng}(\sigma) = \{\sigma(n) : n < |\sigma|\}$. From now on, we think of each $B \subseteq \omega$ as a strictly increasing string $B^\star \in I$, where $B^\star(n)$ is the n-th least element contained in B. For any $\sigma \in I$, we denote by P^σ all supersets of the subset of ω obtained from σ, that is,

$$P^\sigma = \{X \in 2^\omega : \mathrm{rng}(\sigma) \subseteq X\}.$$

Lemma 1. *A set A tt-computes an infinite subset of a Kurtz random set if and only if there exists an infinite set $B \leq_{tt} A$ such that the class $P^{B^\star}$ contains a Kurtz random set. Moreover, if a set A tt-computes an infinite set $B \subseteq \omega$, then $P^{B^\star}$ is a Kurtz null test uniformly relative to A.*

Proof. The first equivalence clearly holds. Assume that there is a truth-table functional Ψ such that $\Psi^A = B$. Inductively define a truth-table functional $\Phi^Z(n)$ by $\Phi^Z(0) = \Psi^Z(0)$, and $\Phi^Z(n+1) = \max\{\Psi^Z(n+1), \Phi^Z(n)+1\}$ for each $n \in \omega$. Then, Φ^Z defines an infinite set $B(Z)$, for every $Z \in 2^\omega$. Moreover, if $B(Z)$ is infinite, then $P^{B(Z)^\star}$ is null. Therefore, $Z \mapsto P^{B(Z)^\star}$ is a uniform Kurtz null test. Hence, $P^{B^\star} = P^{B(A)^\star}$ is a Kurtz null test uniformly relative to A. □

Now, assume that A is low for uniform Kurtz randomness. By Lemma 1, the class $P^{B^\star}$ is a Kurtz null test uniformly relative to A for every $B \leq_{tt} A$. By lowness, $P^{B^\star}$ contains no Kurtz random set. Again by Lemma 1, A *tt*-computes no infinite subset of a Kurtz random set.

(iv)⇒(v): We again use the notation P^f for $f \in I$. As in the proof of Lemma 1, for each increasing total function $f \in I \cap \omega^\omega$ we can see that $f \leq_{tt} A$ if and only if $\mathrm{rng}(f) \leq_{tt} A$. We first recall the following property of P^f.

Lemma 2 (See Greenberg-Miller [10, Theorem 5.2]). *Let $f : \omega \to \omega$ be a strictly increasing function. Then, no Kurtz null test includes P^f if and only if P^f contains a Kurtz random set.*

Proof. Obviously, if P^f contains a Kurtz random set, then there is no Kurtz test including P^f. Conversely, as mentioned in Greenberg-Miller [10, Theorem 5.2], if some nonempty clopen subclass $P^f \cap [\rho]$ is covered by a Kurtz test, then so is all of P^f. Assume that P^f is covered by no Kurtz test. Let $\{Q_n\}_{n\in\omega}$ be a

(non-effective) list of all Kurtz tests. We construct an element $X = \lim_n \xi_n \in P^f$ which is contained in no Kurtz test. Let ξ_0 be the empty string. Assume that ξ_n has been already defined, and it is extendible in P^f. Then, $P^f \cap [\xi_n]$ is not covered by a Kurtz test, as mentioned before. Hence, we can find some ξ_{n+1} extending ξ_n in the class $(P^f \cap [\xi_n]) \setminus Q_n$. Then, $X = \lim_n \xi_n$ is Kurtz random, which is contained in P^f. □

The key notion we will use is that of the *svelte tree* introduced by Greenberg-Miller [10]. A finite antichain $A \subseteq \omega^{<\omega}$ is *k-svelte* via a sequence $\{S_n\}_{n\in\omega}$ of finite sets if

$$S_m \subseteq \omega^{k+m},\ \#S_m \leq 2^m, \text{ and } [\![A]\!] \subseteq \bigcup_{m\in\omega} [\![S_m]\!].$$

Lemma 3 (See Greenberg-Miller [10, Theorem 3.3]). *For a finite antichain $A \subseteq \omega^{<\omega}$ and a natural number $k \in \omega$, if $\mu(\bigcup_{f\in[\![A]\!]} P^f) \leq 2^{-(k+1)}$ holds, then one can find a sequence confirming that A is k-svelte, effectively in A and k.* □

Given a closed set $Q \subseteq 2^\omega$, let $N_Q \subseteq \omega^\omega$ be the set $\{f \in \omega^\omega \cap I : P^f \subseteq Q\}$.

Lemma 4 (See Greenberg-Miller [10, Lemma 4.3 and Remark 4.4]). *If $Q \subseteq 2^\omega$ is clopen, then we can effectively find a finite antichain $A_Q \subseteq \omega^{<\omega}$ such that $N_Q = [\![A_Q]\!]$.* □

We restrict our attention to a bounded subset of N_Q for a given closed set Q. For each order u, we denote by N_Q^u the set of all $f \in N_Q$ such that $|f(n)| = u(n)$ for each $n \in \omega$, where we think of each $f \in N_Q$ as a function from ω into $2^{<\omega}$.

Lemma 5. *Assume that Q is a Kurtz null test. Then, for each order u, there are a computable trace $\{D_n\}_{n\in\omega}$ with bound $n \mapsto 2^n$ and $D_n \subseteq \omega^n$ for each $n \in \omega$ and a computable sequence $\{l_k\}_{k\in\omega}$ of natural numbers such that*

$$N_Q^u \subseteq \bigcup_{n=l_k}^{l_{k+1}-1} [\![D_n]\!], \text{ for every } k \in \omega.$$

Proof. Assume that a Kurtz null test $\{C_n\}_{n\in\omega}$ with $\mu(C_n) \leq 2^{-n}$ and $Q = \bigcap_n C_n$ is given. By Lemma 4, we can effectively find a sequence $\{A_n\}$ of finite antichains generating $\{N_{C_n}\}$. By the definition of N_{C_n}, we have $\bigcup_{g\in[\![A_n]\!]} P^g \subseteq C_n$. Hence, $\mu(\bigcup_{g\in[\![A_n]\!]} P^g) \leq 2^{-n}$. Therefore, by Lemma 3, we can effectively find a sequence $\{S_m^n\}_{m\in\omega}$ confirming that A_{n+1} is n-svelte, uniformly in n. In other words,

$$S_m^n \subseteq \omega^{n+m},\ \#S_m^n \leq 2^m, \text{ and } N_{C_{n+1}} = [\![A_{n+1}]\!] \subseteq \bigcup_{m\in\omega} [\![S_m^n]\!].$$

For each computable order u, because $N_{C_{n+1}}^u$ is compact, it is covered by $\bigcup_{m<c(n)} S_m^n$ for some $c(n) \in \omega$. Note that we can effectively find such a $c(n)$,

since $N^u_{C_{n+1}}$ and $\bigcup_m S^n_m$ are computable. Inductively define $l_0 = 0$, and $l_{n+1} = l_n + c(l_n)$ for each $n \in \omega$. For each $k \in \omega$ and each $n \in [l_k, l_{k+1})$, we define $D_n = S^{l_k}_{n-l_k} \subseteq \omega^n$, where $\#D_n \leq 2^{n-l_k} \leq 2^n$. We now have

$$N^u_Q \subseteq N^u_{C_{l_{k+1}}} \subseteq \bigcup_{m<c(l_k)} [\![S^{l_k}_m]\!] = \bigcup_{n=l_k}^{l_{k+1}-1} [\![D_n]\!]$$

for every $k \in \omega$, as desired. □

Now, we assume that A tt-computes no infinite subset of a Kurtz random set. For each $g \leq_{tt} A$, we claim the existence of a computable trace with bound $n \mapsto 2^{n+1}$ that Kurtz-traces g. Let Ψ be a truth-table functional such that $\Psi(A) = g$, and find a computable order u such that $\Psi(Z \upharpoonright u(n), n)$ is defined for all $n \in \omega$. Then, in particular, $\Psi(A \upharpoonright u(n), n) = g(n)$. Define $f(n) = A \upharpoonright u(n)$ for each $n \in \omega$. By Lemmas 1 and 2, for every order u and every strictly increasing function $f \leq_{tt} A$ with $|f(n)| = u(n)$ for each $n \in \omega$, there is a Kurtz null test $Q \subseteq 2^\omega$ such that $P^f \subseteq Q$ holds. Note that $P^f \subseteq Q$ if and only if $f \in N^u_Q$. Since Q is a Kurtz null test, we have two sequences $\{D_n\}_{n\in\omega}$ and $\{l_k\}_{k\in\omega}$ in Lemma 5. Thus, every $h \in N^u_Q$ is Kurtz traced by $\{D_n\}_{n\in\omega}$ and $\{l_k\}_{k\in\omega}$. For each string $\sigma \in (2^{<\omega})^{<\omega}$, let σ^* denote the last value of σ, that is, $\sigma^* = \sigma(|\sigma|-1)$. Note that $\sigma \in D_{n+1} \subseteq (2^{<\omega})^{n+1}$ implies that $\Psi(\sigma^*, n)$ is defined, since σ^* is of length $u(n)$. For $E_n = \{\Psi(\sigma, n) : \sigma \in D_{n+1}\}$, the trace $\{E_n\}_{n\in\omega}$ Kurtz-traces $n \mapsto \Psi(h(n), n)$ for all $h \in N^u_Q$. In particular, $g : n \mapsto \Psi(f(n), n)$ is Kurtz-traced.

(v)⇒(i): Assume that A is Kurtz tt-traceable via a computable order $n \mapsto 2^{p(n)}$. Given a computable order h, we can find a computable order $u : \omega \to \omega$ such that $h(u(n)) \geq p(n) + n + 1$ for each $n \in \omega$. By our assumption, we have a computable trace $\{D_n\}_{n\in\omega}$ with $\#D_n \leq 2^{p(n)}$ and a strictly increasing computable sequence $\{l_k\}_{k\in\omega}$ of natural numbers, where, for every $k \in \omega$, there is $n \in [l_k, l_{k+1})$ such that $A \upharpoonright u(n) \in D_n$. Without loss of generality, we may assume that $D_n \subseteq 2^{u(n)}$. Then, define $C_k = \bigcup_{n\in[l_k,l_{k+1})} D_n$, for each $k \in \omega$. Note that $A \in [\![C_k]\!]$ for all $k \in \omega$. To estimate the weight of C_k, we note the following inequality:

$$\begin{aligned}\sum_{\sigma\in C_k} 2^{-h(|\sigma|)} &= \sum_{n=l_k}^{l_{k+1}-1} \#D_n \cdot 2^{-h(u(n))} \\ &\leq \sum_{n=l_k}^{l_{k+1}-1} 2^{p(n)-h(u(n))} \leq \sum_{n=l_k}^{l_{k+1}-1} 2^{-n-1} \leq 2^{-l_k} \leq 2^{-k}.\end{aligned}$$

Hence, A is Kurtz h-dimensional measure zero. □

Acknowledgement

The first author was supported by a Grant-in-Aid for JSPS fellows. The second author was supported by GCOE, Kyoto University and JSPS KAKENHI 23740072.

References

1. Barmpalias, G., Miller, J.S., Nies, A.: Randomness notions and partial relativization. Israel Journal of Mathematics pp. 1–26 (2011)
2. Bienvenu, L., Downey, R., Greenberg, N., Nies, A., Turetsky, D.: Characterizing lowness for Demuth randomness, submitted
3. Brattka, V., Hertling, P., Weihrauch, K.: A tutorial on computable analysis. New Computational Paradigms pp. 425–491 (2008)
4. Brattka, V.: Computability over topological structures. In: Cooper, S.B., Goncharov, S.S. (eds.) Computability and Models, pp. 93–136. Kluwer Academic Publishers, New York (2003)
5. Diamondstone, D., Greenberg, N., Turetsky, D.: A van Lambalgen theorem for Demuth randomness, submitted.
6. Downey, R., Griffiths, E.: Schnorr randomness. Journal of Symbolic Logic 69(2), 533–554 (2004)
7. Downey, R., Hirschfeldt, D.R.: Algorithmic Randomness and Complexity. Springer, Berlin (2010)
8. Downey, R.G., Griffiths, E.J., Reid, S.: On Kurtz randomness. Theoretical Computer Science 321, 249–270 (2004)
9. Franklin, J.N.Y., Stephan, F.: Schnorr trivial sets and truth-table reducibility. Journal of Symbolic Logic 75(2), 501–521 (2010)
10. Greenberg, N., Miller, J.S.: Lowness for Kurtz randomness. The Journal of Symbolic Logic 74, 665–678 (2009)
11. Hirschfeldt, D., Nies, A., Stephan, F.: Using random sets as oracles. Journal of the London Mathematical Society 75, 610–622 (2007)
12. Kurtz, S.A.: Randomness and Genericity in the Degrees of Unsolvability. Ph.D. thesis, University of Illinois at Urbana-Champaign (1981)
13. Miyabe, K.: Schnorr trivial and its equivalent notions, submitted
14. Miyabe, K.: Truth-table Schnorr randomness and truth-table reducible randomness. Mathematical Logic Quarterly 57(3), 323–338 (2011)
15. Miyabe, K., Rute, J.: Van Lambalgen's Theorem for uniformly relative Schnorr and computable randomness, to appear in Proceedings of the Twelfth Asian Logic Conference
16. Nies, A.: Lowness properties and randomness. Advances in Mathematics 197, 274–305 (2005)
17. Nies, A.: Computability and Randomness. Oxford University Press, USA (2009)
18. Stephan, F., Yu, L.: Lowness for weakly 1-generic and Kurtz-random. Lecture Notes in Computer Science 3959, 756–764 (2006)
19. Wang, Y.: Randomness and Complexity. Ph.D. thesis, University of Heidelberg (1996)
20. Weihrauch, K.: Computable Analysis: an introduction. Springer, Berlin (2000)
21. Weihrauch, K., Grubba, T.: Elementary Computable Topology. Journal of Universal Computer Science 15(6), 1381–1422 (2009)

A Species Distribution Framework Relying on Membrane Computing with Boundaries

Tamás Mihálydeák[1] and Zoltán Ernő Csajbók[2]

[1] Department of Computer Science, Faculty of Informatics, University of Debrecen
Kassai út 26, H-4028 Debrecen, Hungary
mihalydeak.tamas@inf.unideb.hu
[2] Department of Health Informatics, Faculty of Health, University of Debrecen,
Sóstói út 2-4, H-4400 Nyíregyháza, Hungary
csajbok.zoltan@foh.unideb.hu

Abstract. Recently, modeling of species distributions has emerged into notice. In this paper we propose a twofold integrated and dynamic species distributions framework which simultaneously models geographical space and abstract ecological space. The former corresponds to a P system, while the latter induces a multiset approximation space in the system of multisets attached to P system. Evolution rules model biological events taking place in habitats. Communication rules describe spread of species living in habitats from inside to outwards and/or from outside to inwards. Since biotic interactions act at short distances, execution of communication rules are restricted to membrane boundary zones generated in the multiset approximation space.

Keywords: Species distribution, ecological niche, ecological niche modeling, membrane computing, multiset approximation, membrane boundaries

1 Introduction

Human–caused global environmental change has radically altered the geographical ranges of species widely throughout the world [31, 32]. One of the major component of this phenomenon is the biological invasion. Invasive species are characterized by rapid colonization and uncontrolled spread [27, 30, 10]. Recently, modeling of species distributions has emerged into notice [10]. However, building an integrated species distributions framework which is able to take all the remarkable circumstances and meaningful factors into account dynamically is a difficult task. A brief survey of the modeling techniques of this type can be found, e.g. in [10] and the references therein.

Species distributions can be investigated in *geographical* and abstract *environmental* spaces [21, 14]. The unit of geographical space is the *habitat* where the species and their related organisms subsist. The unit of environmental space is the *ecological niche* which was originated by Grinnell [12] and Elton [8]. Then, we follow the Grinnellian niche concept which covers the concept that "the environmental requirements needed for a species to subsist without immigration" ([14], p. 1373).

In [15], Hutchinson formalized the niche concept as a hypervolume in multidimensional environmental space whose axes represent measurable environmental variables potentially important for species persistence.[3] These variables are both abiotic and biotic [21]. According to the Grinnellian niche concept and its Hutchinson's formalization, a habitat can be thought of as a heap of empty niches which are waiting to be filled [9, 28].

Hutchinson differentiates between the abstract *fundamental (theoretical)* and the *realized (actual)* niche. The former is the full spectrum of environmental factors that can potentially be occupied by a species, whereas the latter is a subset of the fundamental niche which a species actually occupies.[4]

Niches overlap each other which make way for the coexistence of species. Coexistent species interact, and their biotic interactions may radically affect their niches. However, in fact,"biotic interactions act at short distances" ([14], p. 1374). Niches are not stable but dynamic, e.g. their position may be shifted within environmental space [11, 21, 14]. After all, overlapping of niches triggers their differentiation which may result in a stable niche partitioning. Changes of abiotic environmental varibles may affect the niches as well.

Our proposed ecological framework has a twofold structure. Geographical space corresponds to a P *system*, while abstract environmental space induces a *multiset approximation space* in geographical space which joins that P system.

As it is well known, in a P system membranes delimit regions separating "inside" from "outside" [22, 23, 25]. Regions are arranged in a hierarchical structure and are represented by multisets or msets for short [1, 2, 5]. They are endowed with a set of rules providing the functioning of the whole system. There are two types of rules, the evolution and the communication rules. Evolution rules regulate the events taking place in the regions and act on the objects which belong to the region where the actual rules reside. Communication rules prescribe movements and/or exchanges of objects through membranes.

Since biotic interactions act at short distances, in real processes an object actually has to be close enough to a membrane, not necessarily in a spatial sense, in order to be able to pass through it. To model boundary zones around membranes, rough set theory [18, 19] and its generalization, the partial approximation of sets [6, 7] should be a plausible opportunity. However, they work on the bases of the traditional set theory. Thus, to apply their ideas we have to use its improved version worked out for multisets in [17].

The rest of the paper is organized as follows. Section 2 briefly summarizes basic notions of mset approximation spaces. Section 3 defines membrane boundaries in P systems with the help of mset approximation spaces. Section 4 presents the proposed species distribution framework relying on membrane computing with boundaries in detail.

[3] For a detailed review of the historical development and applications of ecological niche modeling, see e.g. [13, 9].

[4] There is a view that "the realized niche is often larger than the fundamental niche" ([26], p. 351).

2 General Multiset Approximation Spaces

2.1 Basic Notations of Multisets

Let U be a finite nonempty set.[5]

Definition 1. *A* multiset M*, or* mset M *for short, over* U *is a mapping* $M : U \to \mathbb{N}$*, where* $\mathbb{N}$ *is the set of natural numbers.*
A set $\mathcal{M}$ *of msets over* U *is called a* macroset $\mathcal{M}$ *over* U.[6]

Definition 2. *Let* M *be an mset over* U.

1. $M^* = \{a \in U \mid M(a) \neq 0\}$ *is called the* carrier (set) *or* support *of* M.
2. *The mset* M *is* simple*, if the cardinality of its support is equal to 1.*
3. M *is the* empty mset*, if* $M^* = \emptyset$*, and denoted by* $\emptyset$.

First, we define the *equality* ($=$) and the *inclusion* ($\sqsubseteq$) relations for msets.

Definition 3. *Let* M_1*,* M_2*,* M*,* M^* *be msets over* U.

1. $M_1 = M_2$*, if* $M_1(a) = M_2(a)$ *for all* $a \in U$*;*
2. $M_1 \sqsubseteq M_2$*, if* $M_1(a) \leq M_2(a)$ *for all* $a \in U$.

Next definitions give the basic multiplicity relations and operations for msets.

Definition 4. *Let* M *be an mset over* U.

1. Multiplicity relation *for the mset* M *is:*
   $a \in M$ $(a \in U)$*, if* $M(a) > 0$*;*
2. n-multiplicity relation *for the mset* M *is:*
   $a \in^n M$ $(a \in U)$*, if* $M(a) = n \in \mathbb{N}$*;*

Definition 5. *Let* M_1 *and* M_2 *be two msets over* U.

1. $(M_1 \sqcap M_2)(a) = \min\{M_1(a), M_2(a)\}$ *for all* $a \in U$ *(*intersection *of* M_1, M_2*).*
2. *For a macroset* $\mathcal{M}$*,* $(\sqcap\mathcal{M})(a) = \min\{M(a) \mid M \in \mathcal{M}\}$ *for all* $a \in U$.
3. $(M_1 \sqcup M_2)(a) = \max\{M_1(a), M_2(a)\}$ *for all* $a \in U$ *(*set–type union *of* M_1, M_2*).*
4. *For a macroset* $\mathcal{M}$*,* $(\bigsqcup\mathcal{M})(a) = \max\{M(a) \mid M \in \mathcal{M}\}$ *for all* $a \in U$.
   By definition, $\bigsqcup\emptyset = \emptyset$.
5. $(M_1 \oplus M_2)(a) = M_1(a) + M_2(a)$ *for all* $a \in U$ *(*sum *of* M_1, M_2*).*
6. n-successive sum *of* M *is* $\oplus_n M$ *(*$n \in \mathbb{N}$*) where* $\oplus_n M$ *is given by the following inductive definition:*
   (a) $\oplus_0 M = \emptyset$*;*
   (b) $\oplus_1 M = M$*;*
   (c) $\oplus_{n+1} M = \oplus_n M \oplus M$.
7. $(M_1 \ominus M_2)(a) = \max\{M_1(a) - M_2(a), 0\}$ *for all* $a \in U$ *(*subtraction M_2 *from* M_1*).*

By the n-successive sum, we define the *n-successive inclusion* relation for msets as follows.

Definition 6. $M_1 \sqsubseteq^n M_2$*, if* $\oplus_n M_1 \sqsubseteq M_2$ *but* $\oplus_{n+1} M_1 \not\sqsubseteq M_2$.

[5] Due to our real–life application, it is sufficient to assume that U is finite, however, it is not necessary from the theoretical point of view.
[6] [16], p. 124.

2.2 Mset Approximation Spaces

Definition 7. *Let U be a nonempty set.*

The ordered tuple $\mathsf{MAS}(U) = \langle \mathcal{M}, \mathfrak{B}, \mathfrak{D}_{\mathfrak{B}}, \mathsf{l}, \mathsf{u} \rangle$ is a general mset approximation space over U, if

1. *$\mathfrak{B}$ is a nonempty macroset, and if $B \in \mathfrak{B}$, then $B \neq \emptyset$; members of $\mathfrak{B}$ is called* base msets*, or* $\mathfrak{B}$-msets *for short.*
2. *$\mathfrak{D}_{\mathfrak{B}}$ is a macroset, and it is an extension of $\mathfrak{B}$ satisfying the following minimal requirements:*
   - *$\emptyset \in \mathfrak{D}_{\mathfrak{B}}$;*
   - *if $B \in \mathfrak{B}$, then $\oplus_n B \in \mathfrak{D}_{\mathfrak{B}}$ $(n = 1, 2, \dots)$.*
3. *$\mathcal{M}$ is a macroset over U in such a way that $\mathfrak{D}_{\mathfrak{B}} \subseteq \mathcal{M}$.*
4. *The maps l, u with domain $\mathcal{M}$ form an approximation pair $\langle \mathsf{l}, \mathsf{u} \rangle$, if*
   *(a) $\mathsf{l}(\mathcal{M}), \mathsf{u}(\mathcal{M}) \subseteq \mathfrak{D}_{\mathfrak{B}}$ (*definability *of l and u);*[7]
   *(b) l and u are monotone, that is for all $M_1, M_2 \in \mathcal{M}$ if $M_1 \sqsubseteq M_2$ then $\mathsf{l}(M_1) \sqsubseteq \mathsf{l}(M_2)$ and $\mathsf{u}(M_1) \sqsubseteq \mathsf{u}(M_2)$ (*monotonicity *of l and u);*
   *(c) $\mathsf{u}(\emptyset) = \emptyset$ (*normality *of u);*
   *(d) if $M \in \mathfrak{D}_{\mathfrak{B}}$, then $\mathsf{l}(M) = M$ (*granularity *of $\mathfrak{D}_{\mathfrak{B}}$);*
   *(e) if $M \in \mathcal{M}$, then $\mathsf{l}(M) \sqsubseteq \mathsf{u}(M)$ (*weak approximation property*).*

   Maps l and u are called the lower *and* upper approximation*, respectively.*

In Definition 7 each condition in *4 (a)–(e)* is independent of the other four. Next proposition summarizes the most basic consequences of Definition 7.

Proposition 1. *Let $\mathsf{MAS}(U)$ be a general mset approximation space over U.*

1. *$\mathsf{l}(\emptyset) = \emptyset$ (*normality *of l).*
2. *$\forall M \in \mathcal{M}\, (\mathsf{l}(\mathsf{l}(M)) = \mathsf{l}(M))$ (*idempotency *of l).*
3. *$M \in \mathfrak{D}_{\mathfrak{B}}$ if and only if $\mathsf{l}(M) = M$.*
4. *$\mathsf{u}(\mathcal{M}) \subseteq \mathsf{l}(\mathcal{M}) = \mathfrak{D}_{\mathfrak{B}}$.*

Definable and crisp msets can be given as usual in rough set theory.

Definition 8. *Let $\mathsf{MAS}(U)$ be a general mset approximation space over U. An mset M over U is* definable *in $\mathsf{MAS}(U)$, if $M \in \mathfrak{D}_{\mathfrak{B}}$, and* crisp *in $\mathsf{MAS}(U)$, if $\mathsf{l}(M) = \mathsf{u}(M)$.*

If an mset $M \in \mathcal{M}$ is crisp, then $\mathsf{l}(M) = \mathsf{u}(M) = M$ by the weak approximation property. Thus, owing to Proposition 1 (3), it is definable as well. However, unlike classic rough set theory, definable msets are not necessarily crisp, because beside $\mathsf{l}(M) = M$, it also possible that $\mathsf{l}(M) \neq \mathsf{u}(M)$.

Definition 9. *Let $\mathsf{MAS}(U)$ be a general mset approximation space over U.*

- *$\mathsf{MAS}(U)$ is a* set–type union *mset approximation space, if $\oplus_n B_1 \sqcup \oplus_k B_2 \in \mathfrak{D}_{\mathfrak{B}}$ for all $B_1, B_2 \in \mathfrak{B}$ and n, k $(n, k = 1, 2, \dots)$;*

[7] As usual, $\mathsf{l}(\mathcal{M})$ and $\mathsf{u}(\mathcal{M})$ denote the range of l and u, respectively.

– $\mathsf{MAS}(U)$ *is a* strict set–type union *mset approximation space, if* $\mathfrak{D}_{\mathfrak{B}}$ *is given by the following inductive definition:*
  1. $\emptyset \in \mathfrak{D}_{\mathfrak{B}}$*;*
  2. $\mathfrak{B} \subseteq \mathfrak{D}_{\mathfrak{B}}$*;*
  3. *if* $\mathfrak{B}^{\oplus} = \{\oplus_n B \mid B \in \mathfrak{B}\ n = 1, 2, \dots\}$ *and* $\mathfrak{B}' \subseteq \mathfrak{B}^{\oplus}$ *, then* $\bigsqcup \mathfrak{B}' \in \mathfrak{D}_{\mathfrak{B}}$*;*

Definition 10. *In a strict set–type union mset approximation space* $\mathsf{MAS}(U)$*,* $\langle \mathsf{l}, \mathsf{u} \rangle$ *pair is a generalized Pawlakian approximation pair, if for any mset* $M \in \mathcal{M}$

1. $\mathsf{l}(M) = \bigsqcup\{\oplus_n B \mid B \in \mathfrak{B}$ *and* $B \sqsubseteq^n M\}$*;*
2. $\mathsf{b}(M) = \bigsqcup\{\oplus_n B \mid B \in \mathfrak{B}, B \sqcap M \neq \emptyset, B \not\sqsubseteq M$ *and* $B \sqcap M \sqsubseteq^n M\}$ *(*boundary *of mset* M*);*
3. $\mathsf{u}(M) = (\mathsf{l}(M) \oplus \mathsf{b}(M)) \ominus (\mathsf{l}(M) \sqcap \mathsf{b}(M))$.

3 *P* Systems with Membrane Boundaries

3.1 On the *P* Systems

We briefly summarize the most important notations concerning P systems.

Definition 11. *A* membrane structure μ *of degree* m *(*$m \geq 1$*) is a rooted tree with* m *nodes identified with the integers* $1, \dots, m$*.*

Later on, in the proposed framework we will work with a membrane structure of degree 2, thus the P system Π in our framework will be of the form as follows.

Definition 12. *Let* μ *be a membrane structure with* 2 *nodes and* U *be a non-empty finite set. The tuple* $\Pi = \langle U, \mu, w_1, w_2, R_1, R_2 \rangle$ *is a* P system*, if*

1. $w_1, w_2 : U \to \mathbb{N}$ *are msets over* U*;*
2. R_1, R_2 *are finite sets of evolution and communication rules in such a way that the mset* w_i *is associated with the set of rules* R_i *(*$i = 1, 2$*).*

Following [24], we define formally the macroset of finite multisets of rules applicable to w_i in the maximally parallel mode as follows:

$$Appl_{\max}(R_i, w_i) = \{R \mid R \in Appl(R_i, w_i), \text{ and there is no } R' \in Appl(R_i, w_i) \text{ with } R' \supsetneqq R\},$$

where $Appl(R_i, w_i)$ denotes a macroset whose members are all multisets of rules from R_i which are applicable to w_i $(i = 1, 2)$.

From $Appl_{\max}(R_1, w_1)$ and $Appl_{\max}(R_2, w_2)$ any maximal multisets of rules can be chosen in order to perform a maximally parallel transition step. Different maximal multisets of rules assign different msets to the same region. Hence, after all, the evolution of the P system Π ramifies in a non-deterministic manner.

3.2 *P* Systems with Membrane Boundaries

The set–theoretic representations of regions are msets, thus boundaries of regions delimited by membranes can be formed in mset approximation spaces. In short, they are also called boundaries of membranes or simply *membrane boundaries*. Then, we can say that an object is close enough to a membrane if it is a member of its boundary. Moreover, inside and outside boundaries of membranes, in other words the closeness to membranes from inside and outside can also be specified. For a more detailed discussion of membrane boundaries we refer to [17].

Remark 1. $\mathfrak{B}$-msets do not comprise a membrane structure because they do not form a hierarchical structure in general.

In the following let $\Pi^0 = \langle U, \mu, w_1^0, w_2^0, R_1, R_2 \rangle$ be an initial P system and $\mathsf{MAS}(\Pi^0) = \langle \mathcal{M}, \mathfrak{B}, \mathfrak{D}_{\mathfrak{B}}, \mathsf{l}, \mathsf{u} \rangle$ with $w_1^0, w_2^0 \in \mathcal{M}$ be a strict set–union type initial mset approximation space with generalized Pawlakian approximation pair.

Definition 13. *Let Π^0 and $\mathsf{MAS}(\Pi^0)$ a P system and its joint mset approximation space as above. If $B \in \mathfrak{B}$, let*

$$N(B,1) = n, \quad \textit{if } B \sqcap w_1^0 \sqsubseteq^n w_1^0;$$
$$N(B,2) = \begin{cases} 0, & \textit{if } B \sqcap w_2^0 = \emptyset \textit{ or } B \sqsubseteq w_2^0; \\ \min\{k, n \mid B \sqcap w_2^0 \sqsubseteq^k w_2^0,\ B \ominus w_2^0 \sqsubseteq^n w_1^0\}, & \textit{otherwise.} \end{cases}$$

Then, for $i = 1, 2$

1. $\mathsf{bnd}(w_i^0) = \bigsqcup\{\oplus_{N(B,i)} B \mid B \in \mathfrak{B}, B \sqcap w_i^0 \neq \emptyset, B \not\sqsubseteq w_i^0\}$ — boundary *of membrane* w_i^0*;*
2. $\mathsf{bnd}^{\mathsf{out}}(w_i^0) = \mathsf{bnd}(w_i^0) \ominus w_i^0$ — outside boundary *of membrane* w_i^0*;*
3. $\mathsf{bnd}^{\mathsf{in}}(w_i^0) = \mathsf{bnd}(w_i^0) \ominus \mathsf{bnd}^{\mathsf{out}}(w_i^0)$ — inside boundary *of membrane* w_i^0.

Informally, $N(B,1) = n$ means how many times the $\mathfrak{B}$-mset B appears in the boundary of w_1^0. In particular, $N(B,1) = 0$ means that B is not in that boundary. $N(B,2) = 0$, if B is not in the boundary of w_2^0; otherwise $N(B,2)$ gives the exact number of how many times B appears in the boundary of w_2^0.

Remark 2. By Definition 13, $\mathsf{bnd}(w_i^0) = \bigsqcup\{\oplus_{N(B,i)} B \mid N(B,i) > 0\}$ $(i = 1, 2)$.

Since $\mathsf{MAS}(\Pi^0)$ is a strict set–union type mset approximation space, the boundary of w_i^0 is definable, i.e. $\mathsf{bnd}(w_i^0) \in \mathfrak{D}_{\mathfrak{B}}$ $(i = 1, 2)$.

Remark 3. With the Pawlakian boundary map b, $\mathsf{b}(w_1^0) = \mathsf{bnd}(w_1^0)$, but $\mathsf{b}(w_2^0) \neq \mathsf{bnd}(w_2^0)$ in general. In the latter case, the Pawlakian boundary of w_2^0 is confined within the region w_1^0. In the former case, however, w_1^0 is a skin region, thus any confinement is unnecessary.

Let us investigate the boundary zone of w_2^0 in more detailed. Having formed the membrane boundary of w_2^0, we constrain the communication rules residing in w_2^0 for its boundary zone $\mathsf{bnd}(w_2^0)$. In order to be in this way, let the execution of a rule $R \in R_2$ define in the following form:

– if a symport rule has the form $\langle u, in\rangle$, it is executed only in the case when $u \sqsubseteq \mathsf{bnd}^{\mathsf{out}}(w_2^0)$;
– if a symport rule has the form $\langle u, out\rangle$, it is executed only in the case when $u \sqsubseteq \mathsf{bnd}^{\mathsf{in}}(w_2^0)$;
– if an antiport rule has the form $\langle u, in; v, out\rangle$, it is executed only in the case when $u \sqsubseteq \mathsf{bnd}^{\mathsf{out}}(w_2^0)$ and $v \sqsubseteq \mathsf{bnd}^{\mathsf{in}}(w_2^0)$.

4 The Species Distribution Framework

Geographical space in which natural vegetation lives corresponds to a P system Π of degree 2. The skin region models a larger area (w_1^0), and the lower neighbor included in the skin membrane models a smaller area (w_2^0).

We basically investigate the dynamic behavior of the lower neighbor w_2^0. Biological laws of species living in w_2^0 are represented by evolution rules from R_2. For instance, they regulate the competition and coexistence of species or their direct interactions with each other.

Spreading of species is modeled by communication rules from R_2. However, biotic interactions act at short distances. Thus we are able to give an account of dynamic exchanges taking place along the boundary of w_2^0. It can be accomplished within mset approximation spaces.

Formally, let $U = \{s_1, s_2, \ldots, s_k\}$ be the set of different species living in the geographical area which is represented by the P system Π. k is the number of different species in the vegetation and embodies diversity, the variety of species, in w_2^0. $w_2^0(s_i) \geq 1$ $(i = 1, \ldots, k)$ is the population size or abundance of the species s_i at a particular moment of time, i.e., it is the number of the specimens of the ith species which are actually present in w_2^0 at that time. Ecological studies require the empirical determination or estimation of population size of plants and/or animals for a defined site and time. There are many various methods of them depending on the features of the investigated species, the habitat, and the available technical means and time [20, 3, 4, 29].

In the Hutchinsonian environmental space, there are numerous theoretical niches which form a connected hypervolume in the multidimensional space respectively. However, they overlap each other in general. Note that the number of ecological niches can be interpreted as the complexity, the variety of ecological processes within a biological community.

It is assumed that there is a one-to-one correspondence between the species s_i's and the fundamental niches in the abstract Hutchinsonian space. Every fundamental niche, in turn, has a realization in the geographical space. Determination or estimation of population size usually concern the number of individuals per a suitable unit of area, e.g., 1 m^2, 1 ha, etc. (of course, it may be a suitable unit of volume, e.g., in hydrobiological research). Let us choose the $\mathfrak{B}$-mset B_i as a simple multiset which contains the species s_i with the empirically determined or estimated multiplicity $B_i(s_i) \geq 1$ per a suitable unit of area.

Every fundamental niche in the abstract Hutchinsonian space is a connected hypervolume whose realization in the geographical space usually consists of dif-

ferent unconnected patches. Different patches, however, may differ in population density. To make the model more realistic, we may choose different $\mathfrak{B}$-msets $B_{i_1}, B_{i_2}, \ldots, B_{i_l}$ for a species s_i, if its population densities differ in different patches. Hence, any patch of a realized niche can be represented as an n-successive sum $\oplus_n B_{i_m}$ with a suitable $n(\geq 1) \in \mathbb{N}$. All B_{i_m} and $\oplus_n B_{i_m}$ are definable msets, i.e., $B_{i_m}, \oplus_n B_{i_m} \in \mathfrak{D}_{\mathfrak{B}}$.

As we mentioned above, overlapping of niches triggers the differentiation of them. These triggers can be linked on to the base msets in a natural way. After all, the differentiation of niches may result in a stable niche partitioning.

The computation process of the proposed twofold species distribution framework follows the general algorithm presented in [17].

It has two *inputs* as defined above:

- an initial P system $\Pi^0 = \langle U, \mu, w_1^0, w_2^0, R_1, R_2 \rangle$ — an observed geographical area with its natural vegetation, and its narrower or wider environment;
- an initial mset approximation space $\mathsf{MAS}(\Pi^0)$ — based on the realized niches which are connected to the species living in the observed area.

The computation process consists of membrane computations in consecutive P systems and trigger activities in consecutive mset approximation spaces. Each iteration is composed of two phases which are completed with additional steps:

1. In the *initialization phase*, the membrane boundary $\mathsf{bnd}(w_2^0)$ is formed within the mset approximation space $\mathsf{MAS}(\Pi^0)$. Then, the scope of communication rules is constrained for the base msets which belong to $\mathsf{bnd}(w_2^0)$.
2. In the *computation phase*, first, the evolution rules in w_2^0 and communication rules in the base msets belonging to $\mathsf{bnd}(w_2^0)$ are executed in a nondeterministic and maximally parallel manner. Evolution rules reflect biological events taking place within w_2^0, while communication rules model the dynamic exchanges of species' specimens along the boundary zone of w_2^0 between $\mathsf{bnd}^{\mathsf{in}}(w_2^0)$ and $\mathsf{bnd}^{\mathsf{out}}(w_2^0)$.

   After the membrane computation halts, a new P system Π' of degree 2 emerges, but the mset approximation space $\mathsf{MAS}(\Pi^0)$ is unchanged.

   However, $\mathsf{bnd}(w_2^0)$ could have changed because, e.g., a base mset may entirely become a part of or get out of w_2^0. If so, $\mathsf{bnd}(w_2^0)$ must be redefined (go to Step 1).

   Otherwise, the dynamic exchanges of species' specimens between $\mathsf{bnd}^{\mathsf{in}}(w_2^0)$ and $\mathsf{bnd}^{\mathsf{out}}(w_2^0)$ start up trigger activities in the base msets belonging to $\mathsf{bnd}(w_2^0)$, which activities spread further throughout the space. Triggers fire in a nondeterministic and a maximally parallel manner.

   If there were some trigger activities, a new mset approximation space $\mathsf{MAS}(\Pi')$ emerges in which $\mathsf{bnd}(w_2^0)$ must be redefined (go to Step 1).

   If there were no any trigger activities, the whole computational process stops. It may happen when the differentiation of overlapped niches results in a stable niche partitioning.

5 Conclusion and Future Work

In this paper we have presented a concise formal framework for describing species distribution. In order to demonstrate our model and clarify its subtle details, biological studies need to be accomplished by interdisciplinary teams.

Acknowledgments

The publication was supported by the TÁMOP–4.2.2.C-11/1/KONV–2012–0001 project. The project has been supported by the European Union, co–financed by the European Social Fund.

The authors are thankful to Ferenc Horváth and the anonymous referees for valuable suggestions.

References

1. Blizard, W.D.: Multiset theory. Notre Dame Journal of Formal Logic 30(1), 36–66 (1989)
2. Blizard, W.D.: The development of multiset theory. Modern Logic 1, 319–352 (1991)
3. Buckland, S., Anderson, D., Burnham, K., Laake, J., Borchers, D., Thomas, L.: Introduction to distance sampling: estimating abundance of biological populations. Oxford University Press, Oxford (2001)
4. Buckland, S., Anderson, D., Burnham, K., Laake, J., Borchers, D., Thomas, L.: Advanced distance sampling. Oxford University Press, Oxford (2004)
5. Calude, C., Păun, G., Rozenberg, G., Salomaa, A. (eds.): Multiset Processing: Mathematical, Computer Science, and Molecular Computing Points of View. Workshop on Multiset Processing, WMP 2000, Curtea de Arges, Romania, August 21-25, 2000, LNCS, vol. 2235. Springer-Verlag, Berlin Heidelberg (2001)
6. Csajbók, Z.: Approximation of sets based on partial covering. Theoretical Computer Science. Special Issue on Rough Sets and Fuzzy Sets in Natural Computing 412(42), 5820–5833 (2011)
7. Csajbók, Z., Mihálydeák, T.: Partial approximative set theory: A generalization of the rough set theory. International Journal of Computer Information Systems and Industrial Management Applications 4, 437–444 (2012)
8. Elton, C.S.: Animal ecology. The Macmillan Company, New York (1927)
9. Fodor, E.: Ecological niche of plant pathogens. Annals of Forest Research 54(1), 3–21 (2011)
10. Gallien, L., Münkemüller, T., Albert, C.H., Boulangeat, I., Thuiller, W.: Predicting potential distributions of invasive species: where to go from here? Diversity and Distributions 16(3), 331–342 (2010)
11. Gause, G.F.: The struggle for existence. Baltimore,The Williams & Wilkins company, (1934)
12. Grinnell, J.: Field tests of theories concerning distributional control. The American Naturalist 51(602), 115–128 (1917)
13. Guisan, A., Thuiller, W.: Predicting species distribution: Offering more than simple habitat models. Ecology Letters 8(9), 993–1009 (2005)

14. Hirzel, A.H., Lay, G.L.: Habitat suitability modelling and niche theory. Journal of Applied Ecology 45(5), 1372–1381 (2008)
15. Hutchinson, G.E.: Concluding Remarks. Cold Spring Harbor Symposia on Quantitative Biology 22, 415–427 (1957)
16. Kudlek, M., Martín-Vide, C., Păun, G.: Toward a formal macroset theory. In: Calude, C., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMP. Lecture Notes in Computer Science, vol. 2235, pp. 123–134. Springer (2000)
17. Mihálydeák, T., Csajbók, Z.E.: Membranes with boundaries. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, G. (eds.) Membrane Computing. 13th International Conference, CMC 2012, Budapest, Hungary, August 28-31, 2012, Revised Selected Papers. LNCS, vol. 7762, pp. 277–294. Springer-Verlag, Berlin Heidelberg (2013)
18. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11(5), 341–356 (1982)
19. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
20. Petrusewicz, K., Macfadyen, A.: Productivity of Terrestrial Animals Principles and Methods. International Biological Programme Handbook No. 13, Blackwell Scientific Publications, Oxford and Edinburgh (1970)
21. Polechová, J., Storch, D.: Ecological niche. In: Jørgensen, S.E., Fath, B.D. (eds.) Encyclopedia of Ecology. vol. 2, pp. 1088–1097. Elsevier, Oxford (2008)
22. Păun, G.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000)
23. Păun, G.: Membrane Computing. An Introduction. Springer-Verlag, Berlin (2002)
24. Păun, G., Rozenberg, G.: An introduction to and an overview of membrane computing. In: Păun et al. [25], pp. 1–27
25. Păun, G., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford Handbooks, Oxford University Press, Inc., New York, NY, USA (2010)
26. Pulliam, H.R.: On the relationship between niche and distribution. Ecology Letters 3(4), 349–361 (2000)
27. Pyšek, P., Richardson, D.M.: Invasive species, environmental change and management, and health. Annual Review of Environment and Resources 35(1), 25–55 (2010)
28. Reich, P., Wright, I., Cavender-Bares, J., Craine, J., Oleksyn, J., Westoby, M., Walters, M.: The evolution of plant functional variation: traits, spectra and strategies. International Journal of Plant Sciences 164(3 Suppl.), S143–S164 (2003)
29. Thomas, L., Buckland, S.T., Rexstad, E.A., Laake, J.L., Strindberg, S., Hedley, S.L., Bishop, J.R., Marques, T.A., Burnham, K.P.: Distance software: design and analysis of distance sampling surveys for estimating population size. Journal of Applied Ecology 47, 5–14 (2010)
30. Thompson, G.D., Robertson, M.P., Webber, B.L., Richardson, D.M., Roux, J.J.L., Wilson, J.R.U.: Predicting the subspecific identity of invasive species using distribution models: Acacia saligna as an example. Diversity and Distributions 17, 1001–1014 (2011)
31. Vitousek, P.M., D'Antonio, C.M., Loope, L.L., M. Rejmánek, M., Westbrooks, R.: Introduced species: a significant component of human-caused global change. New Zealand Journal of Ecology 21(1), 1–16 (1997)
32. Walther, G.R., Roques, A., Hulme, P.E., et al.: Alien species in a warmer world: risks and opportunities. Trends in Ecology and Evolution 24(12), 686–693 (2009)

Undecidability of satisfiability of expansions of FO[<] with a Semilinear Non Regular Predicate over words.

Arthur Milchior

LIAFA, Université Paris 7 - Denis diderot, France.
LACL, UPEC, Créteil, France
Arthur.Milchior@liafa.univ-paris-diderot.fr

Abstract. This paper gives two new characterizations of regular numerical predicates, i.e. sets of integers definable in $\mathsf{FO}[<, \mathrm{mod}]$, and proves that satisfiability of first-order logic over words with an order predicate and a semilinear predicate (i.e. a numerical predicate definable in $\mathsf{FO}[+]$) that is not regular, is undecidable.

1 Introduction

Let mod be the set of modular predicates. Over words, $\mathsf{FO}[<, \mathrm{mod}]$ defines a subclass of the regular languages [1], hence for a given formula we can decide if there exists a word that satisfies it. On the other hand, [2, Lemma 6.3] proved that satisfiability of $\mathsf{FO}[+]$ over words is undecidable. Similarly, [3] prove that $\mathsf{MSO}[<, f]$ is undecidable on $\mathbb{N}$ as soon as f in an increasing function whose image is co-infinite.

In this paper we intend to show that the boundary between decidability and undecidability in $\mathsf{FO}[+]$ is exactly at $\mathsf{FO}[<, \mathrm{mod}]$, and to do this we will need to prove that the theorem of [3] is true even for first-order logic over finite words.

In section 3, Theo. 4 will prove that $\mathsf{FO}[<, f]$ is undecidable as soon as f is some kind of increasing function, and Theo. 5 does the same for $\mathsf{FO}[<, R]$ where R is definable in $\mathsf{FO}[+]$ but not in $\mathsf{FO}[<, \mathrm{mod}]$.

For this, in section 2, we will give two new characterizations of relations definable in $\mathsf{FO}[<, \mathrm{mod}]$, the so-called regular sets, similar to Muchnik's and Michaux-Villemaire's characterizations of $\mathsf{FO}[+]$[4,5], the so-called semilinear sets. Those characterizations may be of independent interest.

The first one, Theo. 1, proves that a set R is definable in $\mathsf{FO}[<, \mathrm{mod}\, k]$ if and only if

1. All sections and diagonals, i.e. hyperplanes defined by $x_i = c$ or $x_i = x_j + c$, of R are definable in $\mathsf{FO}[<, \mathrm{mod}\, k]$ and
2. R can be translated of $-k$ in every direction.

Furthermore by Theo. 1 there is a $\mathsf{FO}[<, R]$ formula that is true over $\mathbb{N}$ if and only if R is definable in $\mathsf{FO}[<, \mathrm{mod}\, k]$.

The second characterization, Theo. 2, states that a semilinear set R is regular if and only if every unary function definable in $\mathsf{FO}[<, R]$ is definable in $\mathsf{FO}[<, \mathrm{mod}]$.

In this paper, we will explain how to construct formulas, but we will not write them explicitly due to lack of space.

1.1 Notation

In this paper, we will represent the r-tuple $(c_1, \ldots, c_r)$ by $\overline{c}$. If k is in integer, $\overline{k}^r$ is the r-tuple composed of r times k. Addition of tuples is defined componentwise, i.e. $\overline{e} = \overline{c} + \overline{d}$ if and only if $e_i = c_i + d_i$ for $1 \leq i \leq r$.

We use standard notation for first order logic with predicate over words, see [6] for example. An alphabet α is a finite set of letters. If X is a set of predicates, $\mathsf{FO}[X]$ over the words of alphabet α is the set of formulas containing every predicate of X applied to variables, an unary predicate $P_a(x)$ for $a \in \alpha$ stating that the xth letter is an a, closed by conjunction, disjunction, negation and by universal and existential first-order quantification. $\exists\mathsf{MSO}[X]$ is the set of formulas beginning by existential second order monadic quantification, followed by a formula of $\mathsf{FO}[X]$. The truth of formula over words is defined the usual way.

We will use the following predicates: "mod" stands for the set of predicates $\{\equiv_k a \mid 0 \leq a < k, a, k \in \mathbb{N}\}$ where $\equiv_k a$ is the predicate that is true for integers equal to a modulo k. The binary relation $\times c$ for $c \in \mathbb{Q}^+$ is true over tuples $(x, y) \in \mathbb{N}^2$ such that $x \times c = y$, and of course $+$ is defined over $\mathbb{N}^3$ as $\{(a, b, c) \mid a + b = c\}$.

For example, let $\alpha = \{a, b\}$, the following formula is true over words with a at even positions and b at odd positions.

$$\forall x, y.\{[(x + 1 = y) \Rightarrow (P_a(x) \Leftrightarrow P_b(y))] \wedge [(\forall z.z \geq x) \Rightarrow P_a(x)]\} \tag{1}$$

2 First-Order Logic and Arithmetic

2.1 Relation with finite models

In this section, we will work with $\mathbb{N}$ as the universe. In particular we will study the set of tuples of integers that verify a given formula.

$t \sim t'$ is used to speak of a predicate of the form $t = t'$, $t < t'$ or $t > t'$. We will use constants and consequently we will always be able to use the predicates $x_i \sim x_j + c$ for every $c \in \mathbb{N}$.

In section 3 we will speak of finite models. Hence we will need to define the notion of *convergence* to state that properties of the formulas over $\mathbb{N}$ can also be used on finite models.

Definition 1 (Function convergence). *Let φ be a formula such that its graph over the canonical model of size n (resp. on $\mathbb{N}$) is a function f_n (resp. f), then we say that φ is converging if the sequence of functions $(f_n)_{n\in\mathbb{N}}$ converges pointwise to f.*

This means that for all $c \in \mathbb{N}$, there exists $m \in \mathbb{N}$ such that for all $n \geq m$, $f_n(c) = f(c)$.

In this paper we will often speak about subsets with a specific form.

We introduce some definitions and notation which deal with $R \subseteq \mathbb{N}^r$.

Definition 2 (Section, diagonal and subspace).

If $R \subseteq \mathbb{N}^r$ then $R^{x_i=c} = \{\overline{x} \in N^{r-1} \mid (x_1, \ldots, x_{i-1}, c, x_i, \ldots, x_r\}$ is called the i*th* section *of R in c and $R^{x_i=x_j+c} = \{\overline{x} \in \mathbb{N}^r \mid (x_1, \ldots, x_{i-1}, x_j + c, x_i, \cdots x_r)\}$ is called the* diagonal *(i, j, c).*

A straight *(resp.* semi-straight*)* subspace *is either R, or a section of (resp. a section or a diagonal of) a straight (resp.semi-straight) subspace of R.*

This means that a straight (resp. semi-straight) subspace of dimension $s \leq r$ of R is defined by $r - s$ equations of the form $x_i = c$ (resp. $x_i = c$ or $x_i = x_j + c$) with $c \in \mathbb{N}$.

We will care about the tuples such that all components are big enough, thus we introduce the following notation.

Definition 3 (l-inside). *Let $l \in \mathbb{N}$, the l-inside of R is $R^{\geq l} \doteq \{\overline{x} - \overline{l}^r \in \mathbb{N}^r \mid \overline{x} \in R, x_i \geq l\}$. This means that $R^{\geq l}$ is the set R moved by $-l$ in every direction, removing the tuples with a coordinate less than l.*

We will also have to assume that the tuples are ordered, hence we will need this new notation.

Definition 4 (Fixed order). *Let E_r be the set of permutations of $[1, r]$. For all $(a_1, \ldots, a_r) \in E_r$ we define $R_{a_1,\ldots,a_r}$ as the restriction of R to r-tuples $(x_1, \ldots, x_r)$ such that $x_{a_i} < x_{a_{i+1}}$ for $1 \leq i < r$. That is*

$$R_{a_1,\ldots,a_r} \doteq R \cap \{\overline{x} \mid x_{a_1} \leq x_{a_2} \leq \cdots \leq x_{a_r}\} \tag{2}$$

Since $R = \bigcup_{\overline{a} \in E_r} R_{\overline{a}}$, we will be able to work separately on each $R_{\overline{a}}$.

2.2 Semilinear, Regular and mod$k$-Regular set

The relations $R \subseteq \mathbb{N}^r$ definable in $\mathsf{FO}[+]$ are called semilinear sets [7] and the ones definable in $\mathsf{FO}[<, \mathrm{mod}]$ are called regular sets [6]. Let $\mathsf{FO}[<, \mathrm{mod}\, k]$ be the restriction of $\mathsf{FO}[<, \mod]$ where only predicate modulo k are used. If R is definable in $\mathsf{FO}[<, \mathrm{mod}\, k]$ we will call it *mod$k$-regular*. Beware that it is different from k-regular sets as defined in [5], which are sets definable in $\mathsf{FO}[+, V_k]$ where $V_k(x)$ is the biggest power of k dividing x, and which are not used in this paper.

Proposition 1 ([7,8]). *Let f be a semilinear unary function. There exist constants $l, m, a_0, b_0, \ldots, a_{m-1}, b_{m-1} \in \mathbb{Q}$ such that $f(x) = a_j x + b_j$ for all $x > l$ congruent to j modulo m.*

An easy adaptation of the proofs of [8] shows that an unary mod$k$-regular function is such that $m = k$ and for $0 \leq j < k$, $a_j \in \{0, 1\}$. In particular for an unary semilinear function f that is not regular there exists $j < k$ such that $a_j \in \mathbb{Q}^+ / \{0, 1\}$.

Presburger proved that for every first-order formula of $\mathsf{FO}[+, <, \mathrm{mod}]$ there is an equivalent quantifier free formula [9]. Another algorithm was given by Cooper [10]. If there is only one free variable, it is even an $\mathsf{FO}[<, \mathrm{mod}\, k]$ formula for some $k \in \mathbb{N}$. The terms used in this formula are sum of variables – where a variable can appear many times in the same term – plus a constant.

A careful observation of [10]'s quantifier elimination algorithm shows that if the original formula is in $\mathsf{FO}[<, \mathrm{mod}\, k]$ then all terms of the quantifier-free formula are a variable plus an integer constant, and the only modular predicate is mod k. Hence we can also assume that every formula in $\mathsf{FO}[<, \mathrm{mod}\, k]$ is equivalent to a quantifier-free formula of $\mathsf{FO}[<, \mathrm{mod}\, k, \{<_a\}_{a\geq 0}]$ where $<_a = \{(x, y) \in \mathbb{N} \mid x < y + a\}$.

We will now give a new characterization of mod$k$-regular relations. We say that a set $R \subseteq \mathbb{N}^r$ is $\overline{k}^r$-periodic if for all $(c_1, \ldots, c_r) \in \mathbb{N}^r$, $\overline{c} \in R$ if and only if $\overline{c} + \overline{k}^r \in R$.

Theorem 1. *Let $R \in \mathbb{N}^r$, and $k \in \mathbb{N}$. The three statements are equivalent.*

1. *R is mod$k$-regular.*
2. *(a) There exists $l \in \mathbb{N}$ such that $R^{\geq l}$ is $\overline{k}^r$-periodic and*
   (b) All sections and diagonals of R are mod$k$-regular.
3. *For every integer $s \leq r$ and every semi-straight subspace H of R of dimension s, there exists l such that $H^{>l}$ is $\overline{k}^s$-periodic.*

Proof. We will show by induction on $r \in \mathbb{N}$ that for every $R \subseteq \mathbb{N}^r$, properties (1), (2) and (3) are equivalent.

If $r = 1$ then properties (2) and (3) are equivalent, and the equivalence with (1) is well-known thanks to quantifier elimination [6]. Let $r > 1$, and we will assume that the equivalences are true for every integer in $[1, r-1]$.

$(1) \Rightarrow (3)$ Let R be a relation defined by $\psi \in \mathsf{FO}[<, \mathrm{mod}\, k]$. It is clear that the straight subspaces H of dimension s defined by $\bigwedge_{j=1}^{s} x_{i_j} = c_j \bigwedge_{j=1}^{t} x_{k_j} = x_{k'_j} + d_j$ are still mod$k$-regular by replacing in ψ every variables x_i by the constant c_j and every x_{k_j} by $x_{k'_j} + d_j$.

The restriction of ψ on H, i.e. the formula ψ where the x_{i_j} are replaced by c_{i_j}, is a boolean combination of formulas of the form $x_i - x_j \sim c$, $x_i \sim c$ with $\sim \in \{<, =, >\}$ and $x_i \equiv_k a$ for $0 \leq a < k$. Let $l = \max\{c \mid x_i \sim c \in \psi\}$.

Then let $\overline{x} \in (\mathbb{N}^{>l})^r$, it is clear that $\overline{x} + \overline{k}^r \in \mathbb{N}^{>l}$. Furthermore every equation $x_i \equiv_k a$ is equivalent to $x_i + k \equiv_k a$, equation $x_i - x_j \sim c$ to $(x_i + k) - (x_j + k) \sim c$, and $x_i \sim c$ to $x_i + k \sim c$ since if $\sim$ is $<$ or $=$ then by definition of l those equations are false and otherwise if $\sim$ is $>$ then the first equation is true, hence the second too. So $\psi(\overline{x}) \Leftrightarrow \psi(\overline{x} + \overline{k}^r)$, and so $H^{>l}$ is $\overline{k}^s$ periodic.

$(3) \Rightarrow (2)$ R is a straight subspace of R hence there exists l such that $R^{\geq l}$ is $\overline{k}^r$-periodic. Its sections and diagonals S satisfy property (3), hence by induction hypothesis there exists l_S such that $S^{>l_S}$ is mod$k$-regular.

(2) $\Rightarrow$ (1) Let R be a relation such that all sections of R are mod$k$-regular and there exists l such that $R^{\geq l}$ is $\overline{k}^r$ periodic. Let $\overline{v}$ be a r-tuple of variables. We will create a formula in $\mathsf{FO}[<, \mathrm{mod}\, k]$ that is true on $\overline{v}$ if and only if $\overline{v} \in R$. It will be the disjunction of two formulas, φ_1 that will consider the tuples having a component smaller than l, and φ_2 the tuples $\overline{x} \in (\mathbb{N}^{>l})^r$.

In the first case, let $\overline{v} \in \mathbb{N}^r$, there exists $1 \leq i \leq r$ such that $v_i < l$. Then $\overline{v}$ is in R if and only if $(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_r)$ is in the section $R^{x_i=v_i}$, and by induction hypothesis, this can be stated with a quantifier-free $\mathsf{FO}[<, \mathrm{mod}\, k]$ formula.

In the second case, we will create a formula to state $\overline{v} \in R^{\geq l}$. If $l > 0$ we can replace every instance of x_i by $x_i + l$ in φ_2, for $i \in [1, r]$ and make a conjunction to verify that all of the x_i are greater than l. Then we can work on $\{\overline{x} \mid \overline{x} + \overline{l}^r \in R\}$ and so assume that $l = 0$.

For every tuple $\overline{v}$ there exists $(a_1, \ldots, a_r) \in E_r$ such that $v_{a_1} \leq v_{a_2} \leq \cdots \leq v_{a_r}$. We will define $\varphi_{a_1,\ldots,a_r}$ the formula that will define $R_{a_1,\ldots,a_r} = R \cap \{\overline{x} \mid x_{a_1} \leq \cdots \leq x_{a_r}\}$. Our formula for R will be the disjunction of all of those formulas.

Without loss of generality we can assume that $v_1 \leq v_2 \leq \cdots \leq v_r$. Let $d = \lfloor v_1/k \rfloor$, and $\overline{y} = \overline{v} - d\overline{k}^r$, then $\overline{v} \in R$ if and only if $\overline{y} \in R$, and furthermore we can see that $0 \leq y_1 < k$.

By property (2), for $a \in \mathbb{N}$, the section $R^{x_1=a}$ is mod$k$-regular, so by induction hypothesis there exists $l(R^{x_1=a})$ such that $R^{\geq l(R^{x_1=a})}$ is $\overline{k}^{r-1}$-periodic. Let $\lambda = \max\{l(R^{x_1=a}) \mid 0 \leq a < k\}$. The formula $\varphi_{1,\ldots,r}$ will be cut in two parts ψ_1 for $v_1 + \lambda < v_2$, and ψ_2 for $v_1 + \lambda \geq v_2$.

We assume that $y_1 + \lambda > y_2$, and let $a = y_2 - y_1$. We have $\overline{y} \in R$ if and only if $(y_2, \ldots, y_r) \in R^{x_2=x_1+a}$, and by property (2) this diagonal is mod$k$-regular.

Otherwise let us assume that $y_1 + \lambda \leq y_2$. Then $\overline{y} \in R$ is equivalent to $(y_2, \ldots, y_r) \in R^{x_1=y_1}$, hence to $(y_2 + dk, \ldots, y_r + dk) \in R^{x_1=a}$, but $(y_2 + dk, \ldots, y_r + dk) = (v_2, \ldots, v_r)$, hence $\overline{v} \in R$ if and only if $(y_1, v_2, v_3, \ldots, v_r) \in R$ where y_1 is the only integer equal to v_1 modulo k and less than k, which can be stated in $\mathsf{FO}[<, \mathrm{mod}\, k]$.

If R is mod$k$-regular then we define $l_k(R)$, and call the lag of R, the smallest integer such that $R^{\geq l_k(R)}$ is $\overline{k}^r$ periodic.

Corollary 1. *For all $r, k \in \mathbb{N}$ there is a formula in $\mathsf{FO}[<, R]$ that does not depends of R that is true if and only if R is mod$k$-regular.*

Proof. Our formula will state the third equivalence of theorem 1. We make a conjunction for every $0 \leq s \leq r$, and every subset E of $[1, r]$ with s elements. Since the number of possible sets is finite, the conjunction is finite. Then we universally quantify the variables x_i for $i \in E$. By fixing those variables we have a straight subspace H. We have to check that there exists a l such that $H^{>l}$ is $\overline{k}^{r-s}$-periodic. We do it by existentially quantifying this l, then by universally quantifying the $r - s$-tuples $\overline{x}$ such that $\lfloor \overline{x} \rfloor \geq l$. Finally we check that $\overline{x}$ is in H if and only if $\overline{x} + \overline{k}^s$ is also in H.

Corollary 1 looks like theorem 2 of [4] for $\varphi \in \mathsf{FO}[<, \mathrm{mod}\, k]$, with $<$ and $\mathrm{mod}\, k$ instead of $+$. The main difference is that [4] has to guess a set of periods while we know that if such a set exists, it must be $\{\overline{k}^r\}$.

2.3 Unary function

The paper [5] proves that $R \subseteq \mathbb{N}^r$ is not semilinear if and only if there is a set $S \subseteq \mathbb{N}$, definable in $\mathsf{FO}[+, R]$ that is not semilinear. In this section, we will prove similar theorems but for semilinear sets R that are not regular and for unary function instead of set of integers. Indeed, if R is semilinear we already know that every set of integers definable in $\mathsf{FO}[<, R]$ is semilinear, hence regular.

Theorem 2. *Let $R \subseteq \mathbb{N}^r$ be a semilinear set that is not regular. There exists a unary function definable in $\mathsf{FO}[<, R]$ that is not regular.*

Lemma 1. *Let $R \subseteq \mathbb{N}^r$ be a semilinear set. There exists an integer $k(R)$ such that every semi-straight subspace $S \subseteq \mathbb{N}$ of R is mod$k$-regular.*

Proof. Let $\psi \in \mathsf{FO}[+, <, \mathrm{mod}]$ be a quantifier-free definition of R.

Let I be a non-empty subset of $[1, r]$, and let $\overline{c} \in \mathbb{N}^r$. Let d_i be $x + c_i$ for $i \in I$, and d_i be c_i otherwise. Then the set $S_{I,\overline{c}} = \{x \mid \overline{d} \in R\}$ is defined by the formula obtained from ψ by replacing x_i by d_i. We can see that the modular predicates used depend only on I and not on $\overline{c}$, hence this set is modk_I-regular for some k_I.

Let $k(R)$ be the least common multiple of the k_I, then every set of integers that is a semi-straight subspace of R is mod$k(R)$-regular.

Proof (of theorem 2). Assume that $R \subseteq \mathbb{N}^r$ is semilinear and not regular. Let ψ be the quantifier-free definition of R in $\mathsf{FO}[+]$, and let $k = k(R)$.

Let $s \leq r$ be the smallest integer such that R has a semi-straight subspace S of dimension s that is not mod$k$-regular. By definition of k and Lemma 1, we have $s > 1$. There is $(a_1, \ldots, a_s) \in E_s$ such that $S_{(a_1,\ldots,a_s)}$ is not mod$k$-regular. Without loss of generality let us assume that it is $(1, \ldots, s)$.

By hypothesis every subspace $S^{x_1=c}$ is mod$k$-regular. Let $l(c)$ be the unary function defined as $l_k(S^{x_1=c})$. By Theo. 1 it is definable in $\mathsf{FO}[R, <]$. We will prove that l is not regular by contradiction. Let us assume that l is regular and let us create a formula $\varphi_R \in \mathsf{FO}[<, \mathrm{mod}]$ defining R.

Since l is regular, there is a q such that l is defined in $\psi \in \mathsf{FO}[<, \mathrm{mod}\, q]$. The function $l'(c) = \max\{l(b) \mid 0 \leq b \leq c\}$ is definable in $\mathsf{FO}[<, l]$ hence in $\mathsf{FO}[<, \mathrm{mod}\, q]$.

By Prop. 1, there exists $c_0, \ldots, c_{q-1}, d \in \mathbb{N}$ and $b_0, \ldots, b_{q-1} \in \{0, 1\}$ such that for every $a > d$ we have $l'(a) = b_e a + c_e$ whenever $a \equiv_q e$.

First, we assume that there exists an $i \in [0, k-1]$ such that b_i equals 0. Since l' is increasing, we have $b_i = 0$ for every $i \in [0, k-1]$, but by hypothesis on $x_1 \leq x_2$ we see that R is included in a finite number of sections of equation $x_1 = b_i$, hence R is regular since the union of a finite number of regular sections is regular. Now we assume that for all i we have $b_i = 1$.

We will make a few modifications to R by removing some semi-straight hyperplanes. We can do it since by hypothesis they are mod$k$-regular, and by translating R - while respecting every constraint that we stated before - so that R become more easy to work with.

By induction hypothesis, hyperplanes of equations $x_1 = b$ with $0 \leq b \leq d$ are mod$k$-regular. We only have to show that $S = R^{\geq l}$ is mod$k$-regular. By working on $R^{\geq l}$ we can assume that $d = 0$.

Let $c = \max\{c_j \mid 0 \leq j < q\}$. The hyperplanes $S^{x_1+b=x_2}$ with $0 \leq b \leq c$ are mod$k$-regular. We only have to show that $T = \{\overline{x} \mid x_1 + c < x_2\} \cap S$ is mod$k$-regular. By translating S by $(0, -c, \ldots, -c)$ we can work on T and assume that $c = 0$.

Since $0 \leq f < k$ the hyperplanes $T^{x_1+f=x_2}$ are mod$k$-regular so there exists m_f such that $T^{x_1+f=x_2}_{>m_f}$ is $\overline{k}^{r-1}$-periodic. Let $m = \max\{m_f \mid 0 \leq f < k\}$. The hyperplanes $T^{x_1=b}$ for $0 \leq b < m$ are mod$k$-regular, so we only have to show that $U = T \cap \{\overline{x} \mid x_1 \geq m\}$ is mod$k$-regular. By translating T by $\overline{-m}^r$ we can work on U and assume that $m_f = m = 0$.

We will prove that with those conditions U is $\overline{k}^r$-periodic, which will imply that U is mod$k$-regular. Let $\overline{c} \in \mathbb{N}^r$, if $c \notin \mathbb{N}_{(1,\ldots,r)}$ then $c \notin U$ by hypothesis, and similarly $c + \overline{k}^r \notin \mathbb{N}_{(1,\ldots,r)}$ so $c + \overline{k}^r \notin U$, so we only have to prove the periodicity for $\overline{c} \in \mathbb{N}_{(1,\ldots,r)}$. Let $u = \lfloor c_1/k \rfloor$, $t = c_1 \bmod k$ and let $\overline{y} = \overline{c} - u(0, k \ldots, k)$. By hypothesis $U^{x_1=c_1,>c_1}$ is $\overline{k}^{r-1}$-periodic so $(\overline{c} \in U) \Leftrightarrow (\overline{y} \in U)$. Let $\overline{z} = \overline{y} + \overline{k}^r$. Since $t \in [0, k-1]$, the diagonal of equation $x_1 + t = x_2$ is $\overline{k}^{r-1}$-periodic, hence we have $(\overline{y} \in U) \Leftrightarrow (\overline{z} \in U)$. Since $U^{x_1=c_1+k,>c_1}$ is also $\overline{k}^{r-1}$-periodic we have $(\overline{z} \in U) \Leftrightarrow (\overline{c} + \overline{k}^r \in U)$, by transitivity $(\overline{c} \in U) \Leftrightarrow (\overline{c} + \overline{k}^r \in U)$.

So U is $\overline{k}^r$-periodic, and by theorem 1, U is mod$k$-regular.

In Theo. 2, we needed R to be semilinear because the construction required for every section to be mod$k$-regular for the same value k. To the best of our knowledge, it is an open question to know if such a function can be created for any non semilinear R.

The definition of the function is not uniform in R because we need to look at every diagonal, and in $\mathsf{FO}[<, \bmod k]$ we can only define a finite number of them.

We also want to state the following theorem. It is similar to [5, Theo. 5.1], but about mod$k$-regular and not about semilinear sets. Since this theorem is not needed in the rest of the paper we will not write the proof, but it is similar to the last one.

Theorem 3. *Let $R \subseteq \mathbb{N}^r$, R is mod$k$-regular if and only if for all $S \subseteq \mathbb{N}$ definable in $\mathsf{FO}[<, R]$, S is mod$k$-regular.*

3 About Satisfiability of Some Class of Formulas Over Words

If φ is a formula without free variable, then we state that φ is finitely satisfiable if there exists a finite model $\mathcal{A}$ such that $\mathcal{A} \models \varphi$. If φ is over the vocabulary of words, then it means that models of φ must be words.

We say that the finite satisfiability of a class of a formula is decidable if there exists an algorithm that decides if a formula of this class is finitely satisfiable. In this section, we will look at what we can tell about satisfiability of formulas over the vocabulary X.

Lemma 2. *Let X be a vocabulary, then $\mathsf{FO}[X]$'s satisfiability is undecidable on words if and only if $\exists\mathsf{MSO}[X]$'s satisfiability is undecidable.*

Proof. Let α be an alphabet and $\varphi \in \mathsf{FO}[X]$ be a formula over words of α^*, let $\varphi' \doteq \exists_{a\in\alpha} P_a\varphi \in \exists\mathsf{MSO}[X]$ is a formula that is satisfiable if and only if φ is satisfiable.

Let $\psi = \exists P_{a_1}, \ldots, P_{a_n}[\forall x \bigvee_{i=1}^{n}(P_{a_i}(x) \wedge \bigwedge_{j=1,j\neq i}^{n} \neg P_{a_j}(x))]\xi$ with $\xi \in \mathsf{FO}[(P_{a_i})_{1\leq i\leq n}, X$, let $\alpha = \{a_i \mid 1 \leq i \leq n\}$ then ξ is a $\mathsf{FO}[X]$ formula over the words of alphabet α, and ξ is satisfiable over words if and only if ψ is satisfiable.

Lemma 3. *Let X be a vocabulary, $\mathsf{FO}[X]$'s satisfiability over words is decidable if and only if $\mathsf{FO}[X, \mathrm{mod}]$'s finite satisfiability is.*

Proof. Let $\varphi \in \mathsf{FO}[X, \mathrm{mod}\, k]$ be a formula over alphabet α. Let $\varphi' \doteq \varphi \wedge 0 \equiv_k 0 \wedge \forall x \neq 0.(\bigvee_{0\leq i<k} x - 1 \equiv_k i \wedge x \equiv_k i + 1)$. Let $\alpha' = \alpha \times [0, k-1]$ and let φ'' be φ' where $P_a(x)$ is replaced by $\bigvee_{i=0}^{k-1} P_{a,i}(x)$ and $x = c \mod k$ by $\bigvee_{a\in\alpha} P_{a,c}(x)$. It is a formula over alphabet α' that is satisfiable if and only if φ is satisfiable.

3.1 Counter Automaton

Let us define k *counter automata* for $k \geq 2$. The paper [11] proves that their halting problem is undecidable. We will be able to prove the undecidability of the satisfiability of $\mathsf{FO}[<, f]$ where f is some kind of increasing function by writing a formula that accepts models that code an halting computation.

Note that [2, Lemma 6.3] reduces the emptiness problem for deterministic linear bounded automaton, which is undecidable, to the satisfiability problem for $\mathsf{FO}[+]$, but they need the full power of the addition to compare the size of two segments, while we can not use the addition.

Given $k \leq 2$, a k-counter automaton is a list of J instructions, and k counters whose value is in $\mathbb{N}$, and initialized at 0. The instructions are "incr(i)" to increment the counter i, "decr(i)" to decrement it, "jmp(m)" to jump at instruction m, "jz(i, m)" to jump at instruction m if the ith counter's value is 0, with $1 \leq i \leq k$, $0 \leq m < J$ and "Halt" to stop the computation. The notation $I(m)$ means the mth instruction. without loss of generality we assume that the only Halt instruction is in position $J - 1$ and that we begin with instruction 0.

Let M be a k-counter automaton, a *configuration* of M is a $(k+1)$-tuple of integers $(m, c_1, \ldots, c_k)$ where m is the index of the next instruction of the automaton and c_j is the value of the jth counter.

A *simulation* of M is a list, finite or infinite, of successive configurations of M. We write $m[l]$, $c_i[l]$ for the lth instruction and the value of the ith counter at time l.

3.2 Undecidability of FO[f,<] For Some Function f

Definition 5 (Increasing enough function). *Let N be an infinite subset of $\mathbb{N}$, $f : N \to N$ be a strictly increasing function such that there exists a sequence of integers $(v_i)_{i\in N}$ with $v_{i+1} = f(v_i)$ and for all i, $([v_i, v_{i+1}] \cap N)$'s cardinality is strictly greater than $([v_i, v_{i-1}] \cap N)$'s cardinality.*

Then f is called Increasing enough function *on N. The set N may be implicit.*

Theorem 4. *Let $N \subset \mathbb{N}$ be definable in $\mathsf{FO}[<, \mathrm{mod}]$, R be a relation such that $f : N \to N$ is an increasing enough function definable in $\mathsf{FO}[<, \mathrm{mod}, R]$. The satisfiability of $\mathsf{FO}[<, R]$ over words is undecidable.*

Proof. We will encode a k-counter automaton M with a formula in $\mathsf{FO}[<, f]$. By lemmas 2 and 3 it is equivalent to work on $\exists\mathsf{MSO}[<, f, \mathrm{mod}]$, so we will use monadic second order variables, V for the v_i's, E for the state and C_i for the ith counter. We must state that those variables are subsets of N. Let us call $[v_n, v_{n+1} - 1] \cap N$ the nth segment. We can check the constraints of V thanks to f.

We encode the nth configuration in the nth segment. We associate with the configuration $(e, c_1, \ldots, c_n)$ a segment S such that the cardinality of $(C_i \cap S)$ is c_i and the cardinality of $(E \cap S)$ is e. The formula $\varphi(M)$ will state that there exists a word that encode a halting computation. Because, even if we can not count, since f is an injection, we can use it to send the nth segment to the $n + 1$th, hence to check that a counter had not changed, had increased or had decreased. We will check that for every x of the nth segment such that $C_i(x)$ is true,then $C_i(f(x))$ is also true, except for the first such x if the ith counter was decremented. We will also check that for each y in the $(n+1)$th segment such that $C_i(y)$ is true, there exists x such that $C_i(x)$ is true and $f(x) = y$, except maybe for the first position such that there are no such x if the ith counter is incremented.

We see that for v, v' and v'' such that $[v, v' - 1]$ and $[v', v'' - 1]$ are two consecutive segments, we can use f to assert that the number of times a predicate is true in segment $[v, v'[$ is equal (less or minus 1) to the number of times it is true in $[v', v''[$, and clearly we can also assert that this number is equal to a constant. Hence, we only have to see that this is enough to write formulas that state that the transition between two segments is correct, that the first and last segment encode an initial state and an accepting state, respectively.

Corollary 2. *$\mathsf{FO}[<, \times c]$ for $c \in \mathbb{Q}^+/\{0, 1\}$ and $\mathsf{FO}[+]$ are undecidable.*

3.3 Undecidability of FO[R,<] for R Semilinear Not Regular

Theorem 5. *Let R be a semilinear relation, then either R is definable in $\mathsf{FO}[<, \mathrm{mod}]$, or the satisfiability of $\mathsf{FO}[R, <]$ is undecidable over words.*

Proof. If R is definable in $\mathsf{FO}[\mathrm{mod}\, k, <]$ then the satifisfiability of $\exists\mathsf{MSO}(\mathrm{mod}\, k, <, R)$ is decidable, since we can replace R by its definition, and apply the algorithm to decide satisfiability of $\exists\mathsf{MSO}(\mathrm{mod}\, k, <)$ [1].

Let us assume that R is not regular. By Theo. 2 we can define a semilinear converging function $f \in \mathsf{FO}[<, R]$ that is not regular. Hence by Theo. 4 there is a d such that for all $n \in \mathbb{N}$ there is $m(n)$, k and $a < k$ such as $x \equiv_k a$ and $d \leq x \leq m(n)$ we have $f(x) = bx + c$ with $b \in \mathbb{Q}^+/\{0, 1\}$. If $b < 1$ we can use the inverse of this function.

We then have an increasing function, hence by Theo. 4 $\exists\mathsf{MSO}[<, R]$ is undecidable.

So the satisfiability of $\mathsf{FO}[<, R]$ is undecidable.

The condition of semilinearity is mandatory, it is used in the proof to construct the increasing enough function in Theo. 2. Indeed let $T = \{2^k \mid k \in \mathbb{N}\}$, T is clearly not semilinear but [12] proved that $\mathsf{MSO}[+, T]$ is decidable. Similarly [13] proved that $\mathsf{MSO}[<, \lceil \sin(x) \rceil]$ is decidable while the graph of $\lceil \sin(x) \rceil$ is not semilinear.

4 Conclusion

We have proved that the satisfiability of $\mathsf{FO}[<, R]$ is undecidable on words for every semilinear relation R that is not regular. Let Π_i be the set of first-order formulas with $i-1$ alternation of quantifiers, beginning with universal ones. A careful study of the formula – not here by lack of space – allows to prove that undecidability already holds for $\Pi_2[<, +1, R]$ and $\Pi_3[<, R]$.

We see many directions for further research. We may want to minimize the number of alternations of quantifiers needed to have undecidability, and conversely to find an algorithm to decide those logics with one or two alternations. We plan to study the expressive power of $\mathsf{FO}[+1, f]$ where f is an uninterpreted function.

We also intend to extend Theo. 4 to use $+1$ instead of $<$ – the proof is a little bit more complex –, we also want to have less restrictive conditions on f, to accept strictly increasing function with an infinity of n such that $f(n+1) > f(n)+1$, as in [3], or an f such that there is an infinite number of integers n such that $f^{-1}(n)$ is infinite.

I would like to thanks the referees for providing useful corrections to the first version of the paper. I also thank Alexis Bes, who was my advisor during the internship when I did this research.

References

1. Büchi: Weak second-order arithmetic and finite automata. Zeitschrift für Mathematische Logik und Grudlagen der Mathematik **6** (1960) 66–92
2. Lange, K.J.: Some results on majority quantifiers over words. In: IEEE Conference on Computational Complexity, IEEE Computer Society (2004) 123–129
3. Thomas, W.: A note on undecidable extensions of monadic second order successor arithmetic. Archiv für mathematische Logik und Grundlagenforschung **17** (1975) 43–44
4. Muchnik, A.A.: The definable criterion for definability in presburger arithmetic and its applications. Theor. Comput. Sci. **290**(3) (2003) 1433–1444
5. Michaux, C., Villemaire, R.: Presburger arithmetic and recognizability of sets of natural numbers by automata: New proofs of cobham's and semenov's theorems. Ann. Pure Appl. Logic **77**(3) (1996) 251–277
6. Straubing, H.: Finite Automata, Formal Logic, and Circuit Complexity. (1994)
7. Ginsburg, S., Spanier, E.H.: Semigroups, presburger formulas, and languages. Pacific Journal of Mathematics **16** (1966) 285–296
8. Finkel, A., Leroux, J.: Presburger functions are piecewise linear. Research Report LSV-08-08, Laboratoire Spécification et Vérification, ENS Cachan, France (March 2008) 9 pages.
9. Presburger, M.: On the Completeness of a Certain System of Arithmetic of Integers in which Addition is the Only Operation. (1929)
10. Cooper, D.C.: Theorem proving in arithmetic without multiplication. Machine Intelligence 7 (1972) 91–99
11. Minsky, M., Laboratory, L.: Recursive Unsolvability of Post's Problem of "tag": And Other Topics in Theory of Truing Machines. Group report. Massachusetts Institute of Technology, Lincoln Laboratory (1960)
12. Elgot, C.C., Rabin, M.O.: Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. J. Symb. Log. **31**(2) (1966) 169–181
13. Semenov, A.L.: Logical theories of one-place functions on the set of natural numbers. Mathematics of the USSR - Izvestia **22** (1984) 587–618

PASSIVE COMPUTATION AND THE POWER OF INACTIVITY

HOW TO GET A BRAIN TO COMPUTE WITHOUT FIRING ITS NEURONS

Bernard Molyneux

The University of California at Davis

Abstract. Passive computation is neural computation that is implemented using timely non-activations at key junctures. Consider e.g. an 'inverted granny neuron' that fires all the time except when granny is around, whose sudden *non-firing* signifies granny's presence. In this paper, I explore the limits of passive computation by showing how any neural net could be transformed into one that, for an arbitrary input, processes using only passive computation, i.e. without firing any neurons. I suggest that this undermines the assumption, dominant in the mind sciences, that *activity* is required for consciousness.

1 Introduction

David Chalmers ([CE4], p.298) asks "an interesting question", namely:

> ... whether active causation is required for experience. Could a thermostat have experience when it is sitting in a constant state (in a sense "causing" an output, but without really doing anything)? Or does it have experience only when in a state of flux?... I do not know the answer to this question, but there is an intuition that some sort of activity is required for experience.

The claim is usually presented more strongly: That not only is activity needed, but, specifically, that *neural* or *brain* activity is required. Hence we have Antony ([CE1], p. 109) claiming that 'the state of one's unused brain parts at a time is no more relevant to the character of one's experience at that time than is the state of the moon'; and Maudlin ([CE7], p. 409) insisting that, since conscious experiences occur at particular times, the 'nearly inescapable' conclusion is that 'one's phenomenal state at a time is determined entirely by one's brain activity at that time'. Neuroscientists almost universally agree with the philosophers in making these assumptions. For Crick and Koch ([CE5], p. 475), for example, the 'key question... [is] how do the neural *activities* that correspond to consciousness... differ from somewhat similar brain *activities* that are unconscious?' (my italics in this and all of the following). For Melloni et al ([CE8], p. 2858) it is to discover 'which signatures of *neural activity* critically differentiate conscious from

unconscious *processing*'. And according to Dehaene and Naccache ([CE6], p. 3), it is to determine 'whether there is a systematic form of information *processing* and a reproducible class of *neuronal activation patterns* that systematically distinguish mental states that subjects label as 'conscious' from other states.'

I claim, though, that human-like mentality, including consciousness, can emerge from a neural system that is entirely inactive. I won't have room to present the entirety of my case, but I can provide the foundational argument in which I show that, for any sequence S of stimuli lasting as long as you like, it is possible to transform any neural net N that would react to S into one that (i) reacts to S, and to all other stimuli, just as N does but which (ii) exhibits no internal neural activity in processing S. The key insight is that neural nets are capable of what I call *passive computation* – that is, computation that is performed by sets of neurons *not* firing at appropriate times. If this neural net, which could be as complex as the human brain, were in control of a body, it would control it the way a regular brain would even while none of the internal neurons were firing.

2 A Functionally Equivalent But Inactive Brain

2.1 Passive Processing

Let's warm up with a thought experiment:

Example 1. The Passive Signaling System. Settlers living at the center of an island construct an advance warning system to tell them when raiders approach their coasts. Sentinels flash a coastal beacon on the quarter-hour if the horizon is clear, but do nothing if enemies approach (or if they are slain). Each intermediary beacon, between the coast and the settlement, flashes just when a more peripheral beacon is lit, transmitting a signal inland. When all is well, the settlers see lights from every direction upon the quarter-hour. But when there is danger a signal *absence* indicates that their enemies approach, and from which direction. If that happens, the settlers ready themselves for battle.

The story encodes several lessons. Firstly, that there is such a thing as passive *representation*, for the non-lighting *signifies* or *encodes* the approach of enemies. Secondly, that passivity can *cascade*. For when some sentinel fails to flash, this results in a series of resultant non-flashings heading inland. Thirdly, it even points the way towards *passive computation*: Suppose an inland sentinel can see two coastal beacons, and he stays dark upon the quarter-hour if and only if *both* the beacons he sees stay dark. He therefore implements a passive AND-gate, silently communicating the approach of enemies from two directions. Another sentinel might function as a passive OR-gate, similarly; and another might perform a passive NOT operation, staying dark if and only if he sees a flash.

In principle a similar passive representation and computation system could be implemented in an organic system. A pain signal, for example, could consist,

not in a sudden burst of activation against a backdrop of inactivity, but in the sudden absence of an otherwise constant message to the effect that everything is fine.

2.2 Neural Nets

Fig. 1 shows the basic idea behind a neural net. Each node, at a given time, takes one of two possible values; 0 or 1. It will take value 1 if the totality of incoming stimulation exceeds its 'threshold'. If it takes value 1, it is said to 'activate' or 'fire'. 0 corresponds, similarly, to inactivity.

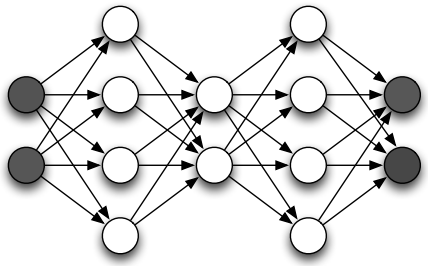


Fig. 1. A typical neural net. The shaded nodes implement input and output.

When a node 'fires', it activates 'outgoing' connections (which, in the diagram, point away from it; 'incoming' connections point towards it). Those connections implement a linear function $wy + x$, where w is the 'weight' of the connection, y is the value (either 0 or 1) of the connection's 'source' node (the node to which its tail connects) and x is the 'baseline' activity of the connection. The connections will, in turn, stimulate the nodes downstream.

The input to the net consists of the values in the 'input nodes' (shaded nodes without incoming arrows), which correspond to what would be sensory receptors in an organic net. The output of the net consists of the values in the output nodes (the shaded nodes without outgoing arrows) which, in an animal, would control muscle contractions. We refer to all other nodes – that are not input or output nodes – as 'internal nodes'.

To claim that all the nodes can be made to take value zero while retaining functional equivalence would be to ignore the fact that functional equivalence requires the same inputs (in the input nodes) to be mapped to the same outputs (in the output nodes). Hence the input and output nodes must retain their old (nonzero) values. The claim may only be that any net can be changed so that, for a special input, it processes that input without anything activating *between* input and output.

2.3 Nullification

A neural net is 'nullified' with respect to a stream S of stimuli if it is replaced by a functionally equivalent net that processes S without activating any internal neurons or connections. Surprisingly, any neural net can be nullified with respect to any one stream S of stimuli. We will first show this for just those nets that do not use any neuron twice in the processing of S, generalizing to other nets later. We proceed in two passes, first silencing the connections and then the nodes:

Phase 1: Take any internal node n in the net. Let 'v_i' denote the activity of the ith input connection c_i on S. Take the function $wy + x$ that c_i implements and replace it with $wy + x - v_i$. It will now transmit v_i less in all situations, and hence transmit 0 on S. After doing this to all connections feeding into n, reduce the threshold of n by V, where V is the sum over the v_i. The total effect of all these changes is that n will receive V less stimulation in all circumstances, but will require V less stimulation in order to fire. Functionally speaking, we have performed a total-of-nothing operation, yet we have succeeded in making the connections that feed into n transmit no activation for stimulus S.

Note that when we 'zero', in this way, the connections heading into a node n, but compensate by changing n's threshold, all other nodes are unaffected. It follows, then, that we can perform the same procedure on another node, and another, without undoing our earlier work. Each time we do it, we produce a new, functionally equivalent, net, but one where fewer connections activate in processing stimulus stream S. After performing the operation on all the neurons in the net, the net as a whole, though functionally equivalent to the original net, is such that none of its connections fire when S is processed.

Phase 2: Now take any internal node in the (modified) net. Check whether it activates when the net as a whole is exposed to S. If not, leave it be. But if so, 'invert' it, so that instead of firing when the sum of its inputs is *greater* than its threshold θ, it fires when the total stimulation is *less than or equal to* θ. To functionally compensate, replace the linear function on each outgoing connection from $wy + x$ to $-wy + x + w$ (i.e. replace the weight w with $-w$ and replace the baseline $+x$ with $+x + w$).[1] To see how this cancels out the changes made to the node itself, suppose that the original neuron was a 'granny neuron' that fired when and only when granny was around, so that:

> Before the change, when granny was around, the former neuron would fire, hence y would equal 1, and so an arbitrary connection exiting the original granny neuron would transmit $z = 1w + x$ – i.e. $w + x$. After the change, when granny is around, the inverted neuron receives the same input as before. But this now causes it *not* to fire. Hence y equals 0. But since the arbitrary outgoing connection has been altered to implement

[1] This may require some signals to be negative in value. A negative signal is simply one that has inhibitory force. One way to implement such things is to have signals with wavelengths, and to have the phase angle between stimulatory and inhibitory signals be equal to π radians.

the function $-wy+x+w$, the inverted neuron transmits $z = -w0+x+w$ which equals $w + x$. The same as before.

Before the change, when granny was *not* around, the former neuron would *not* fire, hence y would equal 0, and so an arbitrary connection exiting the original granny neuron would transmit $z = 0w + x$ which equals x. After the change, when granny is not around, the inverted neuron *fires*, hence y equals 1, hence the (altered) arbitrary connection transmits $z = -1w + x + w$ which equals x. The same.

The operation just described satisfies the following two conditions: (i) If n is the node upon which we are operating then, after the operation, n is disposed to take value 0 when the net as a whole is exposed to S (as just shown). (ii) For all η, where η is some other node or a connection in the net, if, before the operation, η was disposed to take value 0 when the net as a whole was exposed to S then, after the operation, η remains disposed to take value 0 when the net as a whole is exposed to S.

In other words, the operation *deactivates* one node while *preserving* the deactivation of all other internal nodes and connections in the net. To see that the operation has property (ii), divide the entities in the net into two groups:

I: The connections that immediately exit n.

II: All other internal nodes and connections.

The elements of group I were specifically altered so as to always transmit the same activation after the operation that they did beforehand. Hence, if they transmitted 0 in response to S before, they will transmit 0 afterwards. With respect to all other nodes, we simply note that nothing has changed. In more detail, an arbitrary node or connection η in group II can change its activation only by either changing its function or by receiving different input. But it does not calculate a different function, since the operation on n altered η in no way. And we specifically ensured that the changes to n made no difference to its outputs, so they can't have made any difference to the inputs received by η.

Since the zeroing of each node preserves the earlier zeroings, we can zero node after node while preserving our work. When we perform this operation on all the internal nodes in the net, they will all take value 0 for stimulus S.

2.4 Causation

It's puzzling to see how an inactive brain could *cause* any behavior in the body or vehicle it is assigned to control. Inactivity, intuitively, ought to beget more inactivity, and hence an inactive net ought to cause no movement in the corresponding body. It's tempting to think that this nixes the notion that an inactive brain could maintain functionally equivalence. Yet, unless we have made a mistake, functional equivalence has been established. So how is the same behavior *caused* when the inactive brain can't *cause* anything?

To see how, let the output nodes be neuromuscular junctions that cause contractions or twitches in a muscle sheet when they take nonzero activation. Now take an arbitrary neuromuscular junction 'o' (for 'output'). Initially, the

net for which o is an output node is one that activates throughout in response to S, passing activation onto o in the regular fashion. We will show that the activity value of o is preserved, in all conditions including S, as we nullify the net for S.

To see how, note that, at each step in the nullification, one of three things happens:

(i) o itself is modified. Or...

(ii) the connections leading directly to o are modified. Or...

(iii) some other connection or node is modified.

Proceeding in reverse: In case (iii), where some connection or node k other than o or its incoming connections is modified, this makes no difference to o. k was changed, after all, in such a way that it had the same effect on the rest of the net.

What about case (ii), where some connection c leading to o is modified? That happens in one of two conditions: The first is when c's source node is zeroed and c has to compensate. But when c is made to compensate, it is ipso facto made to send the exact same value it used to. So no change to o results. The other occasion is when, along with all of o's other incoming connections, c is zeroed. When that happens, o itself is modified to compensate. But in that case, o is modified in such a way as to ensure that it takes the same value as before. Specifically, it has its threshold lowered so that, if the connections now feed it V less input in all circumstances, it requires V less activation to fire. And thus it continues to take the same values that it did before any change.[2]

Situation (i), then, is already taken care of. For we only modify o itself when we are compensating for the changes in its incoming connections. And as we just saw, this results in o taking the same value it always did.

Thus the nullification of the net leaves o's activity levels, across all possible circumstances, unchanged. Which means that if o activated before, and hence caused a twitch in the muscle sheet, then it activates afterwards likewise.

2.5 More Complex Nets

We have been assuming that the net-to-be-nullified only grants each node and connection in the net one opportunity to fire during the processing of S. But what if a net has connections that feed back into earlier layers? Or if the stimulus stream is temporally extended? Then the same neuron might fire twice in the processing of S.

In such cases, we cannot straightforwardly apply the techniques outlined above. For if a node has more than one opportunity to activate, then it might take different values on each. Inverting it, in that case, would simply switch one activation for another, not silence it for the whole of input S. Similarly, lowering the activation of a connection by a single quantity v across all circumstances

[2] Indeed, the short answer to how it is that o can activate when it receives input zero is that it has had its threshold lowered so that it requires zero stimulation to fire. It fires now *by default*, unless it is inhibited.

might not nullify the connection tout court, since it might take different values v and v' on different occasions. Since there is no single shift in absolute values that will transform both v and v' into 0 for input stream S, this net cannot be nullified for the input stream S. At least, we cannot nullify it by merely using the techniques above.

How, then, might we extend our techniques? Essentially, we must (i) transform the net into one that only grants each node and connection one opportunity to fire for the duration of the stimulus S and then (ii) use the procedures above. Hence the transformation occurs in two passes:

(i) Expansion: We replace each internal neuron n by a set of special turn-taking neurons $f_1(n)$, $f_2(n)$, $f_3(n)$, etc, each of which is disposed to remain inactive until it is its turn to play the role of n. So $f_1(n)$ plays the role of n for the first time-step, $f_2(n)$, plays that role for the second, and so on. The collection is not infinite, so ultimately $f_1(n)$ will take over again, and the cycle will reinitiate. We expand the brain until the collection of turn-taking neurons is so large that no node, and no connection between them, fires twice in the processing of S. See fig. 2.

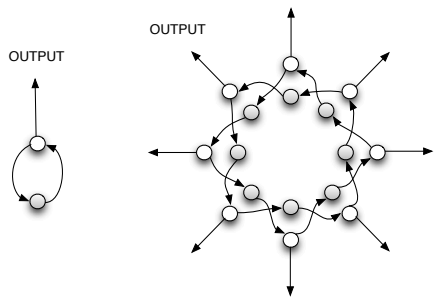


Fig. 2. Each node in the simple net on the left is replaced with a looping track of turn-taking nodes on the right. In the net on the left, if the shaded neuron fires at time t_n, it will cause the unshaded node to fire at time t_{n+1}. Equivalently, in the net on the right, the shaded node whose turn it is at time t_n will, if it fires at time t_n, cause the unshaded node whose turn it is at t_{n+1} to fire at t_{n+1}.

(ii) Nullification: After expansion, each turn-taking node can fire a maximum of one time in the processing of S. We now nullify for S using the procedures above.[3]

[3] It's not correct to say that an inverted turn-taking neuron fires until it is switched off, for a turn-taking neuron keeps quiet until its turn. Rather, it activates by default when its turn comes unless it is subject to inhibition.

3 Functional Equivalence

The net at the end of this procedure is, to all appearances, very different from the net we started out with. In fact, though, the resultant net is functionally equivalent to the original net at the level of individual neural firings. To see why, forget pass (ii), nullification, for the moment and just consider the expanded net $E(N)$ we get from the original net N after pass (i), expansion. For any arbitrary stimulus stream S, each firing at an arbitrary node n at time t in the simpler net N corresponds to a firing with the exact same value, causing the exact same output in its efferent connections and causing identical valued firing events downstream, at node $f_t(n)$ in the more complex net $E(N)$. And vice versa. Hence there is two-way functional equivalence between the two nets at the level of firing events. The only reason the more complex net *looks* architecturally different is because it has been 'unfolded' with respect to time, with a set of turn-taking neurons in the latter net replacing a single neuron in the former. But this difference is a difference in the identity of the pieces that are arranged to play the implemented roles, not a difference in the implemented roles themselves, and hence not a functional difference.

All that changes when we nullify this net is that the corresponding values between n at t and $f_t(n)$ are not always identical values. Indeed, sometimes $f_t(n)$ uses a 0 to accomplish what n at t did with a nonzero value. Still, what is important is that the new encoding is functionally equivalent to the old, and we know that it is from the fact that, when we nullified the net, we *ensured* that each neuron, and each value taken by each node or connection, played the same local (and hence global) role. Nullification, after all, was a process that proceeded locally, node by node, with the functional integrity of each neuron, connection and value being preserved at every step.

What we can now see is that any brain can, with respect to any sequence of stimuli S no matter how long or complex, be reengineered so that it processes S passively. We could, to work with a memorable example, create a brain that passively computes the stimuli that Lee Harvey Oswald encountered when interrogated by the Dallas police, and which passively maps those inputs to the exact behavioral responses that Oswald himself exhibited. Of course, it would only passively compute that input if it were arrested for assassinating the US President and asked those exact questions in that exact order, in a room of the exact right color etc, and that is vanishingly unlikely. But the point is one of principle: That if it *were* put in that situation it would exhibit the same behaviors exhibited by Oswald, but without any of its neurons firing.

Oswald's actual interrogation ran to several hours. To get the system to passively compute for that long, we must simply loop together sufficiently many Oswald-nets (in the manner of figure 2) so that no node or connection activates twice for that period, then nullify the resulting net. Such a network would obviously be gargantuan, too large for a human head, and so we might have to put it on the moon and have it radio-control its body. But these are, again, practical problems, distinct from the fundamental points of principle.

Could the creature be conscious even while its brain remains inactive? Is the passively produced behavior perforce unintelligent? Many philosophical considerations intrude upon our answering these questions, too many to contemplate fully here. Still, we can make a brief and broad case to the effect that the inactive brain is mindful. Two separate arguments can be brought to bear:

(A) Recall that the system is functionally equivalent to the original system at the level of neuron firings. Since functionalists generally believe that the interesting functional facts are implemented by the neurofunctional facts they ought to believe that, in reproducing the neurofunctionality, we thereby produce a system with similar mental states. Non-functionalists, meanwhile, may add additional requirements - e.g. that the brain be made of the right stuff (e.g. flesh, not copper wires and silicon). Block [CE3], for example, persuasively deploys the "Chinese Nation" thought experiment to argue that getting the functionality right isn't enough. Ironically, though, since the point we have made is functional, there is no obvious reason we should not be able to meet any additional requirements that people like Block might add, on an ad-hoc basis if need be. For example, we can build our silly net out of organic tissue, if that is required.

(B) That the system behaves just like a regular, intelligent human, even when its brain is deactivated. If one maintains that it does this despite having no (conscious) mental states, then one appear to admit that (conscious) mental states play no fundamental role in the production of regular, intelligent, behavior. Holding such a position, though, has historically proved problematic.

4 Possible Worries

I have little space to respond fully to all possible worries. But there follows some prominent concerns with some sketched answers.

The idea that inactivity can encode something interesting is not an entirely new idea. If the temporal coding view of neural information is correct then spike trains are to be thought of as sequences of 0s and 1s, with the 0s implemented by non-firings. One might still think, however, that 0s get their informational relevance by punctuating sequences of 1s – in other words, inactivity becomes informational by interrupting streams of activity. What the current project shows, in contrast, is that inactivity has content even when there is no activity for it to punctuate. For it is the embedding of the inactivity against a backdrop of counterfactual, not temporal, activity that renders the inactivity informational.

One might suppose that nullification merely produces a system that codes the null response as the correct response to one particular input (and maps the null response, in turn, to an appropriate bodily response). But this would be neither new nor distinctive to connectionists nets. However, we could have done things a lot more easily if that were all we were after. We could have imagined a stimulus-stream S that switches the brain off and a corresponding pre-program in the muscles that reacts as-if-to-S when the brain switches off. But what would correspond, on this arrangement, to the inverted granny neuron, passively representing granny's presence by *not* firing? Presumably, nothing.

Yet without such representation we have no concrete reason to believe that the inactive net is conscious. In contrast, because we showed that the nullified net is equivalent to the original net at the level of neural firing opportunities, we know that the new granny node (or track of nodes) continues to play the same functional role. It merely does so passively. So granny-representation is still going on in our nullified net. And granny consciousness with it.

Alternatively, one might worry that gargantuan systems are able to fake intelligence (and perhaps other mental attributes) – e.g. by using giant look-up tables consisting of every conversation that is (say) less than an hour long [CE2]. Is our gargantuan, expanded, nullified net a gargantuan cheat in the same way? To check, note how such intelligence-faking systems are excluded by compactness criteria such as that offered by Shieber [CE9], according to which a system counts as intelligent by having a competence for passing Turing tests of length logarithmic to its storage capacity. So if the storage space of the system is large, the length of the Turing test it can pass must be correspondingly large (though 'large' is relative. In fact, a conversation-memorizer could be unmasked by a Turing test lasting about a minute, according to Shieber's estimates, even if its storage capacity were equal to the whole universe.) This foils the possibility of giant look-up tables by ensuring that the test we give the system will be longer than any pre-canned conversation it could have in memory. Does it also count our expanded nullified net as being an intelligence faker? No. For it is perfectly functionally equivalent to the human brain with which we started, so it must have the same competence for conversation that the human has. In other words, it can continue indefinitely.

References

[CE1] Antony, M.: Against Functionalist Theories of Consciousness. Mind and Language 9 (2) (1994)105-23.

[CE2] Block, N. Psychologism and Behaviorism. Philosophical Review XC (1) (1981) 5-43.

[CE3] Block, N. Troubles with Functionalism. Minnesota Studies in the Philosophy of Science 9 (1978) 261-325

[CE4] Chalmers, D.: The Conscious Mind. Oxford University Press (1982) 315–333.

[CE5] Crick, F. Koch, C.: What are the neural correlates of consciousness? in Problems in Systems Neuroscience, eds. Jan Leonard Hemmen, Terrence Joseph Sejnowski, Oxford University Press US (2006) .

[CE6] Dehaene, *S*. Naccache, L.: Towards a cognitive neuroscience of consciousness: basic evidence and a workspace framework. Cognition 79 (2001) 1-37.

[CE7] Maudlin, T.: Computation and Consciousness. Journal of Philosophy 86 8 (1989) 407-432.

[CE8] Melloni, L., Molina, C., Pena, M., Torres,D., Singer, W., Rodriguez, E.: Cognitive Synchronization of Neural Activity across Cortical Areas Correlates with Conscious Perception. The Journal of Neuroscience, March 14, 27(11) (2007) 2858 2865.

[CE9] Shieber, *S*.: The Turing Test as Interactive Proof. Nous, 41 (4) (2007) 686 - 713

Formal Philosophy and Legal Reasoning: The validity of legal inferences

Clayton Peterson*

Department of Philosophy, Université de Montréal, Canada
clayton.peterson@umontreal.ca

Abstract. The aim of the present paper is to introduce a formal logic that can help to test the validity of legal inferences. We begin by presenting the rationale of our method and then we expose the philosophical foundations of our analysis. If formal philosophy is to be of help to the legal discourse, then it must first reflect upon the law's fundamental characteristics that should be taken into account. Our analysis shows that the (Canadian) legal discourse possesses three fundamental characteristics which ought to be considered if one wants to represent the formal structure of legal arguments. These characteristics are the presupposed consistency of the legal discourse, the fact that there is a hierarchy between norms and obligations to preserve this consistency and the fact that legal inferences are subjected to the principle of deontic consequences. We present a formal deontic logic which is built according to these characteristics and provide the completeness results.

Key words
Legal obligations, Legal discourse, Normative consistency, Deontic consequence, Non krikpean semantics

1 Introduction

Deontic logic began with the work of von Wright [34] and has since been interpreted in many different ways. Although the approaches that we find nowadays within the literature vary significantly, most all share a common feature: the use of possible world semantics and the interpretation of deontic logic as a modal logic.[1] Among these approaches the reader will find monadic, dyadic, temporal, non-monotonic, first-order, dynamic and *stit* deontic logics (see [23] for an overview).[2] The modal interpretations are usually characterized by the fact that

* The author would like to thank Jean-Pierre Marquis for his valuable comments on a previous draft of this paper and for his continuous support. This paper was supported by the Social Sciences and Humanities Research Council of Canada.

[1] Note that there are some approaches that do not use possible world semantics, as for instance the input-output logic of [19].

[2] See for example [4] for an introduction to monadic and dyadic deontic logic, [30] for an overview of dyadic deontic logic, [31, 32] for temporal deontic logic and [22] for non monotonic deontic logic. See [14] for an introduction to *stit* logics and [37] for a sketch of possible world semantics in deontic logic.

the truth value of a normative proposition depends upon the truth value of the descriptive proposition in the scope of the deontic operator. These interpretations are of the *Ougth-to-Be* type, meaning that a deontic proposition $O\varphi$ expresses that a specific description φ of the world *ought to be* the case, or that the world *ought to be* in a specific state. However, one can nonetheless find some modal interpretations of the *Ought-to-Do* type, which are characterized by the fact that the proposition in the scope of a deontic operator is instead interpreted as the name of an action. This kind of interpretation can be found notably within the dynamic approach to deontic logic[3], where an action is forbidden when its performance implies that the world is in a state within which there is a violation V. It can also be found in deontic logics based upon a boolean algebra[4].

The present paper does not aim at criticizing these approaches but rather aims at providing a new one, based on different philosophical foundations. The idea is to propose an interpretation of deontic logic which is based upon the analysis of (Canadian) legal norms and the fundamental principles that guide their interpretation. Our goal is not to provide an analysis of how we use the 'ordinary deontic language' or how the 'ordinary deontic reasonings' work, as Castañeda [9, p.38] would say, but is rather to show how a normative reasoning *should* work. The objective is to establish some basic properties that govern a 'correct' use of a (legal) normative inference. In section 2, we expose the rationale behind our framework and present the philosophical foundations of our system. The formal deontic logic is presented in section 3. We conclude in section 4 by presenting the limitations of our approach. Completeness results are provided within Appendix I together with a semi-formal method which can be used to analyze the validity of legal inferences in Appendix II.

2 Philosophical assumptions

Contra the modal interpretation of deontic logic, we do not consider deontic propositions as being of the *Ought-to-Be* type. Following Solt [29, p.350], the truth value of a normative proposition $O\varphi$ for the actual world does not depend upon the truth value of φ at any 'deontic alternative'. The truth value of a deontic proposition $O\varphi$ does not rely on the performance value of φ in every accessible 'deontically perfect world'. The fact that φ is obligatory depends upon the existence of a norm, which is established by some authority [1, pp.97,102] and aims to guide one's actions [36, p.134]. This is consistent with the legal adage *nullum crimen sine lege*: there is no crime without law, nor any obligation without a norm. Therefore, the truth value of a normative proposition $O\varphi$ does not depend upon the truth value of the descriptive proposition φ in the scope of the deontic operator but depends upon the fact that there is a norm which makes φ obligatory. By the same reasoning, the truth value of a deontic proposition

[3] See [20, 21] for the introduction of dynamic deontic logic and, among others, see [25], [6], [15], [27], [3], [12], [28] or [24].
[4] See for instance [26].

does not depend upon the fact that there is a 'violation' in every state where the action is performed.

We do not wish to analyze legal obligations within the framework of modal logics since the operator O_i does not behave as a normal K modality of type $\Box_i$. Although O_i marks the property of an action, this operator is not interpreted as a predicate since it can be applied to combinations of actions, which are expressed by molecular compound of descriptive formulas. But O_i is not interpreted as a modality of type $\Box_i$ since we do not wish to obtain formulas such as $O_i(A \vee \neg A)$ or $O_i A \supset O_i(B \vee \neg B)$, which are dubious from a legal point of view.

It is noteworthy that we are not considering the legal discourse as a *normative system*, that is a "set of agents (human or artificial) whose interactions can fruitfully be regarded as norm-governed [7, p.265]". This point is important: we are not trying to *describe* how agents interact within a normative system. Rather, we wish to define a semantical consequence relation for the validity of normative inferences. Our approach is *normative* rather than *descriptive*. It concerns the analysis of legal reasoning from a critical point of view. This is mainly why we will not use *stit* or dynamic logic: we are not focusing on the notion of *agency*. Rather, we are analyzing the semantical consequence relation within a legal argument.

According to the semantical dichotomy between facts and norms [cf. 18], descriptive and normative propositions are not true in the same conditions. While a descriptive proposition is true or false in regards to the world it describes, the truth of a normative propositions (which says how the world should be or how people should act, rather than how the world is) depends upon the existence of a norm, which is established by some authority. We distinguish between a norm and a normative proposition: the former is neither true nor false while the latter can be true or false (the truth value of the normative proposition depending upon the existence of a norm). A normative proposition expresses that an action (either a specific action or a class of specific actions) possesses a deontic property. For example, from the Canadian Criminal Code we can conclude that the action 'stealing a red bicycle' possesses the propriety 'forbidden' or, equivalently, that the negation action 'not stealing a red bicycle' possesses the property 'obligatory'. Assuming that an action possesses a deontic property, we want to develop a basic logic that can represent how this property can be transmitted from an action to another.

One property of a legal discourse is that there are no obligations without norms. Following Chellas [10, p.24], we want to be able to represent situations where there are no obligations, and thus our system must not include 'absolute' or 'unconditional' obligations. Likewise, following Jones and Pörn [16, p.279], it is always possible for someone to act against one's obligations, hence tautologies are not obligations unless there is a norm that makes it so. Formally, we do not want our system to validate theorems of the form $\vdash O_i \top$ or $\vdash A \supset O_i \top$ (with $\top$ a tautology). Also, since the meaning of a deontic operator changes when it is iterated [16, p.286], it follows that, according to Castañeda [9, p.66], it must not be possible to iterate the same deontic operator. For now, our system will concen-

trate on propositions within which there is only one type of deontic operator (one authority) which cannot be iterated. Finally, we will not be considering mixed propositions (i.e., propositions composed with both descriptive and normative atoms). Indeed, it is unclear that a material conditional is sufficient to represent the transmission of truth value between descriptive and normative propositions since the truth value assignment for a descriptive proposition differs from the truth value assignment of a normative one. The connective '$\supset$' does not preserve truth from a descriptive proposition to a normative one. For example, it is possible to have a situation where $p \supset P_i q$ is true but $(p \wedge p') \supset P_i q$ is false. Therefore, we will not be considering mixed formulas for now and will concentrate only on normative ones.[5]

The deontic logic we propose is based upon an analysis of the Canadian legal discourse and the fundamental principles that guide its interpretation. Since legal norms are meant to guide one's actions, it follows that norms must be consistent since it would be impossible to act accordingly with an inconsistent set of norms. Thus, we assume the criteria of *normative consistency*: obligations are supposed to be consistent (at least after interpretation) since the legislator is presupposed to be rational, meaning that he is presupposed to think rationally and logically [11, p.387]. The set of norms created by the legislator is therefore presupposed to be consistent, and thus legal obligations must not be interpreted as contradictory. Consistency is a rational criteria that ensures the accessibility, the authority and the equity of the law [11, p.387]. The Canadian legal discourse is considered as forming a 'logical system' where there is *horizontal* and *vertical* consistency, meaning that obligations from a same set of norms are consistent and that the different sets of norms are consistent with each other [11, p.388]. In other words, the set of all legal laws is presupposed to form a consistent whole [11, p.433]. Consistency is a rational criterion that enables one to judge the value of a set of norms, which can be examined through the consistency of the set of obligations that it generates.

The relation of hierarchy between norms is meant to preserve the legal system's consistency. On the one hand, hierarchy is necessary to insure the *vertical* and *horizontal* consistency. Since there can be (*a priori*) contradictions between different sets of laws or different laws within a set of norms, hierarchy enables us to preserve the consistency of the whole system in resolving potential conflicts of obligations.[6] On the other hand, laws are constructed in an hierarchical manner. For example, in Canada, no legislation can go against the Canadian Constitution. The fact is that there are sets of laws that are superior to others by construction, although these relations of hierarchy can also be explicitly mentioned in the law itself [11, p.45 and p.450].

[5] For a more detailed analysis of these points, see [23].

[6] It is noteworthy that introducing a relation of hierarchy between norms and obligations to solve a conflict might create another. In this case, the system will have to interpreted in a way that preserves its consistency, perhaps by introducing other hierarchies.

A legal reasoning is characterized by the fact that it is hypothetical: a legal conclusion cannot be drawn without a legal premise. This is consistent with the semantical dichotomy. A valid legal inference implies that there is a set of hypothetical norms (or obligations) from which legal conclusions are drawn. Thus, an obligation is conditional to a set of norms (or principles), which is established by some authority. The fact that obligations are derived from norms (which are stated by certain authorities) implies that normative inferences are ruled by a specific criteria of validity. Since there is a finite number of norms, it would seem that there is also a finite number of obligations that will derive directly from these norms. For example, we can derive from the law that it is forbidden to steal. However, norms are often formulated in order to be applied to a class of specific actions. The obligation that derives directly from the law is to *not steal*, but clearly it is meant to be applied to a class of actions: not stealing a car, not stealing money, not stealing Paul's money, not stealing Peter's car, not stealing Paul's car and Peter's bicycle, etc. In other words, there is a much greater number of obligations that can be derived from a specific norm. But how do we legitimately conclude them since they are not *explicitly* mentioned by the norms?

To answer this question, we follow Alchourrón and Bulygin [1, p.102] and distinguish between *fixed* and *derived* obligations. A fixed obligation stems from a norm and can be general in order to be applied to a class of specific actions. Thus, there will be a finite number of fixed obligation, according to the norms which entails them. A derived obligation can be inferred from a fixed one. From *it is forbidden to steal* we can derive *Paul has the obligation to not steal Peter's money*. The rule which governs this type of inference is the *principle of deontic consequences* [8, p.13][7]: if A is an obligation and A implies B, then B is also an obligation. The mere formulation of the law implies that there are derived obligations. In *French Civil Law* it is obvious since the law is formulated in a general way and applies implicitly to each particular case that falls within its scope.[8] Even though it seems to be the contrary in *Common Law*, where the law is constructed upon each particular judgment, the same principle applies nonetheless since a judgment applies to a class of particular actions. Even if a judgment comes from a particular action, the case law applies to other similar cases. Put differently, there are actions that are obligatory (or forbidden) even though they are not mentioned explicitly by the law (or the case law). The law is not an enumeration of all possible cases. The law says that it is forbidden to steal. What it means is that it is forbidden to do any action that implies stealing.[9] Normative consistency implies that legal obligations are subjected to the principle of deontic consequences since it is possible to deduce from the law

[7] See also [33, p.421].

[8] Quebec's legislation is composed of both French Civil Law and Common Law.

[9] We are not pretending that logic can resolve the problem of interpreting the law. We only say that laws are formulated in order to include different kinds of actions, which has nothing to do with deciding whether the action falls within the scope of the law or not.

certain things that logically follow, assuming that the legislator is rational [11, p.422]. The aim of the present paper is to define this consequence relation.

To sum up, the analysis of the Canadian legal discourse brought to light three fundamental characteristics, namely its presupposed consistency, the fact that hierarchy is meant to preserve that consistency and the validity of the principle of deontic consequences, which enables one to infer derived obligations from fixed ones. If one is to apply formal philosophy to law, then one must take these characteristics into account.

3 The logic of legal obligations

According to these characteristics, the logic we propose relies upon a consequence relation which is defined in function of normative consistency and deontic consequences. For now, we do not need to include hierarchy within the formal definition since we are only considering one type of deontic operator which cannot be iterated. However, as we will see, hierarchy will play a role for the analysis of the soundness of an argument (see Appendix II). The basic idea of our approach is to define a consequence relation which can represent how the property 'obligatory' can be transmitted from an action (or a combination of actions) to another. We follow Castañeda [9, p.46] and assume a distinction between different types of obligations, where the type of an obligation depends upon the set of norms from which it can be derived [1, p.120].

3.1 Syntax

The formulas of OL are built from $\mathcal{L}$ and have the form O_iA, where O_i is a single type of obligation and A is a proposition that refers to an action or a combination of actions.

$$\mathcal{L} = \{(,), Prop, \neg, \supset, O_i\}$$

$Prop = \{p_1, \ldots, p_n, \ldots\}$ is a denumerable set of propositional descriptive atoms. Descriptive propositions are understood as descriptions of *actions*, hence not as *any* description. The other logical connectives $\wedge$, $\vee$, $\equiv$, F_i and P_i are defined as usual, with

$$F_iA =_{def} O_i\neg A \qquad \text{(Interdiction)}$$
$$P_iA =_{def} \neg O_i\neg A \qquad \text{(Weak permission)}$$

We use A, B and C as meta-variables. Well-formed formulas of $\mathcal{L}$ ($WFF_{\mathcal{L}}$) are defined recursively by:

$$p_i \in WFF_{PL} \text{ for all } p_i \in Prop \qquad (1)$$
$$\text{if } A, B \in WFF_{PL}, \text{ then } \neg A,\ A \supset B \in WFF_{PL} \qquad (2)$$
$$\text{if } A \in WFF_{PL}, \text{ then } O_iA \in WFF_{OL} \qquad (3)$$
$$\text{if } A, B \in WFF_{OL}, \text{ then } \neg A,\ A \supset B \in WFF_{OL} \qquad (4)$$
$$\text{if } A \in WFF_{PL} \text{ or } A \in WFF_{OL}, \text{ then } A \in WFF_{\mathcal{L}} \qquad (5)$$

PL stands for propositional logic and OL for the logic of legal obligations. Thus defined, $WFF_{\mathcal{L}}$ does not contain any mixed formulas or any formula where there is an iteration of a deontic operator. This language incorporates the dichotomy between the descriptive and the normative. The axiom schema for normative consistency is represented by (A1), which is propositionally equivalent to the axiom (D) of standard deontic logic.

$$\neg(O_i A \wedge O_i \neg A) \tag{A1}$$

The principle of deontic consequences is represented by the rule of inference (R1) and its use is restricted by two conditions.

$$\begin{array}{l|ll} 1 & \vdots & \\ 2 & O_i A_1 \wedge \cdots \wedge O_i A_n & \\ 3 & (A_1 \wedge \cdots \wedge A_n) \supset B & \\ 4 & O_i B & \text{R1 l.2,3} \end{array} \tag{R1}$$

First, $\{A_1, \ldots, A_n\}$ cannot be the empty set ($n \geq 1$), otherwise B would be a theorem of PL and there would be a set of absolute obligations. Since every obligation is conditional to a norm, it follows that B must be a consequence that depends upon $\{A_1, \ldots, A_n\}$. Hence, B must not be a tautology. Secondly, the conditional

$$(A_1 \wedge \cdots \wedge A_n) \supset B$$

must be either a theorem of PL or an hypothesis. As a result, (R1) cannot be applied to *any* material conditional but can only be applied to a conditional which is either a theorem of PL or a hypothesis. Let us note that since (R1) can be applied to an hypothesis, it follows that this rule is invalid in a standard system such as KD. We introduce OL in a natural deduction system. We say that A is a theorem of OL, written $\vdash_{OL} A$, when there is a proof of A without the use of any hypothesis. In addition to (R1), we assume the rules of PL [cf. 13, p.35], that is

1. hypothesis (H);
2. reiteration (Reit);
3. detachment ($\supset$ *out*);
4. conditional proof ($\supset$ *in*);
5. double negation (DN).

Although OL contains only normative propositions, the proofs of the theorems are done in $\mathcal{L}$. Thus constructed, OL (and moreover $\mathcal{L}$) behaves at the propositional level according to the rules of PL. We write 'PL' as a justification in a proof when the formula is a theorem of PL.[10]

[10] One the one hand, it should be noted that our syntactical system is similar to that of [34] (see the axiomatization in [35]). However, the two are not equivalent, since

3.2 Semantics

Let $Act = \{\Gamma_A : \Gamma_A \text{ is a positive action}\}$ be a denumerable set of *positive* actions (e.g. walking, talking, stealing, ...), where Γ_A stands for the action described by the proposition A, and $\overline{Act} = \{\overline{\Gamma}_A : \overline{\Gamma}_A \text{ is a negative action}\}$ a denumerable set of *negative* actions (e.g. not walking, not talking, not stealing, ...). The set $\mathbb{A} = Act \cup \overline{Act}$ is thus the denumerable set of all possible actions (actions in Act do not need to be atomic). Each action in $\mathbb{A}$ can be described by a proposition of WFF_{PL}. Let the sequence $s^\alpha =< s_1^\alpha, ..., s_n^\alpha, ... >$ be an arbitrary enumeration of Act and the sequence $\overline{s}^\alpha =< \overline{s}_1^\alpha, ..., \overline{s}_n^\alpha, ... >$ an enumeration of $\overline{Act}$, where $\overline{s}_i^\alpha$ refers to the negation of the action s_i^α. A propositional variable p_i (or a molecular compound A_i) refers to an action s_i^α member of an arbitrary sequence s^α, while $\neg p_i$ (or $\neg A_i$) refers to $\overline{s}_i^\alpha$. If A_i refers to s_i^α, then s_i^α is the action Γ_{A_i} (described by A_i), that is the i^{th} member of s^α. An arbitrary sequence s^α (and its counterpart $\overline{s}^\alpha$) is an interpretation of the language $\mathcal{L}$. It assigns a descriptive proposition of WFF_{PL} to each object of the domain (i.e., each action).

In order to formalize semantically the principle of deontic consequences, we assume that the set $\mathbb{A}$ is pre-ordered by '$\sqsubseteq$'. For example, if we suppose that

$$p_1 = \text{Peter steals a red bicycle}$$
$$p_2 = \text{Peter steals a bicycle}$$
$$p_3 = \text{Peter steals}$$

then we have $\Gamma_{p1} \sqsubseteq \Gamma_{p2} \sqsubseteq \Gamma_{p3}$. This allows us to give a semantical account of the relation of implication between actions. If an action Γ_A implies another action Γ_B, then $\Gamma_A \sqsubseteq \Gamma_B$. Thus, if we assume that 'stealing implies not violating the law' is true in some (descriptive) interpretation, then we assume that 'stealing' $\sqsubseteq$ 'not violating the law' in some normative model. Let $\mathcal{M}$ by a standard model of PL. In oder words, if $\models_{\mathcal{M}} p_1 \supset p_2$ (i.e., $p_1 \supset p_2$ is assumed to be true in $\mathcal{M}$), then $\Gamma_{p_1} \sqsubseteq \Gamma_{p_2}$ holds in the normative interpretation. (We do not assume the converse because it would lead us to ideality. It is not because an entailment between actions is assumed in a normative interpretation that it is necessarily true in the descriptive one. For instance, one can assume that 'it is obligatory that *if one is in a public place, then one is not naked*' holds in a normative model while the conditional in the scope of the operator is false in a descriptive one.)

Let $\mathcal{N} = \langle W, \mathbb{A}, \sqsubseteq, a\rangle$ be a normative model, where $W \neq \varnothing$ is the universe of discourse and contains normative propositions (which are members of WFF_{OL}), $\mathbb{A}$ is a denumerable set of actions pre-ordered by '$\sqsubseteq$' and $a : W \longrightarrow \{\top, \bot\}$ a function which assigns truth values to propositions in W. Let $\mathcal{O}$ be a proper subset of $\mathbb{A}$ ($\mathcal{O} \subsetneq \mathbb{A}$). Informally, $\mathcal{O}$ is a set of actions which have the property 'obligatory', meaning that $\mathcal{O}$ is the extension of the concept 'obligatory' within

$\nvdash_{OL} O_i(A \supset A) \supset O_i(A \vee \neg A)$. On the other hand, even if (R1) may look like the Rule 0 in [2], they differ from the important fact that $\{A_1, ..., A_n\}$ cannot be the empty set. Also, his syntactical system is equivalent to KD, which is not equivalent to OL since (R1) is invalid in the standard system.

a normative interpretation. The set has to satisfy three conditions:

$$\text{If } \Gamma_A \in \mathcal{O}, \text{ then } \overline{\Gamma}_A \notin \mathcal{O} \tag{C1}$$

$$\text{If } \Gamma_A \in \mathcal{O}, \text{ then } \Gamma_B \in \mathcal{O} \text{ for any } \Gamma_A \sqsubseteq \Gamma_B \text{ such that } \not\models_{PL} B \tag{C2}$$

$$\text{If } \Gamma_A \in \mathcal{O} \text{ and } \Gamma_A \sqsubseteq \Gamma_{B \supset C}, \text{ then } \Gamma_B \sqsubseteq \Gamma_C \tag{C3}$$

The first condition assures the consistency of $\mathcal{O}$ (i.e., normative consistency) and the second implies that $\mathcal{O}$ is closed 'upwards' when B is not a tautology of the (classical) propositional calculus. The third condition says that if the action described by A is semantically linked to the action described by $B \supset C$, then the action described by B is semantically linked to the action described by C insofar as the action described by A is obligatory. C2 and C3 allow us to represent the principle of deontic consequences. For a normative interpretation $\mathcal{N}$, $a_{\mathcal{N}}(A) = \top$ if and only if there is a sequence $s^\alpha =< s_1^\alpha, ..., s_n^\alpha, ... >$ which satisfies A. We now define satisfaction recursively.

Definition 1. *For any A, s^α* satisfies *A if and only if*

1. *If A is $O_i B$, then*
   (a) *If B is p_i, then s^α satisfies A iff s^α satisfies p_i iff $s_i^\alpha \in \mathcal{O}$ (i.e. the i^{th} member of s^α is a member of $\mathcal{O}$).*
   (b) *If B is $\neg C$, then s^α satisfies A iff $\overline{s}^\alpha$ satisfies C.*
      i. *$\overline{s}^\alpha$ satisfies p_i iff $\overline{s}_i^\alpha \in \mathcal{O}$*
      ii. *$\overline{s}^\alpha$ satisfies $\neg D$ iff $\overline{\overline{s}}^\alpha = s^\alpha$ satisfies D*
      iii. *$\overline{s}^\alpha$ satisfies $D \supset E$ iff s^α satisfies D and $\overline{s}^\alpha$ satisfies E*
   (c) *If B is $C \supset D$, then s^α satisfies A iff either*
      i. *s^α satisfies $\Gamma_m = C \supset D$ (i.e. $s_m^\alpha \in \mathcal{O}$) or*
      ii. *s^α satisfies D (provided that Γ_C is not Γ_D).*[11]
2. *If A is $\neg B$, then s^α satisfies A iff s^α does not satisfy B.*
3. *If A is $B \supset C$, then s^α satisfies A iff s^α does not satisfy B or s^α satisfies C.*[12]

We say that a normative proposition A is *valid* when it is true for any normative interpretation, i.e., $\models A \Leftrightarrow \forall \mathcal{N}, \models_{\mathcal{N}} A$. This system is proven to be sound and complete (see proof in Appendix I). The reader may note that the deduction theorem does not need to be proven in a natural deduction system since it follows immediately from the conditional proof rule. See Appendix II for a graphical representation of the system and how it can be applied to analyze legal inferences.

[11] Informally, the first condition means that the conditional represents a fixed obligation while the second represents a derived obligation.

[12] Let us note that it would be incorrect to infer from the fact that $\overline{s}^\alpha$ satisfies p_i that it also satisfies $A \supset p_i$ from condition 1(c). Rather, if $\overline{s}^\alpha$ satisfies p_i, then s^α satisfies $\neg p_i$, and thus s^α satisfies $A \supset \neg p_i$.

4 Closing remarks

This paper contributes to the literature insofar as it provides a simple language that covers a portion of the intuitive validity of legal inferences which was not covered by other frameworks in the literature. To sum up, we introduced a formal deontic logic which can be applied to analyze the validity of legal reasoning. It is our view that if formal philosophy is to be of help to the legal discourse, then it must first reflect upon the law's fundamental characteristics that should be taken into account. We provided the reader with a brief analysis of the Canadian legal discourse and we exposed three fundamental characteristics which ought to be considered if one wants to represent the formal structure of legal arguments. These characteristics are the presupposed consistency of the legal discourse, the fact that there is a hierarchy between norms and obligations to preserve this consistency and the fact that legal inferences are subjected to the principle of deontic consequences. The formal logic that was built according these characteristics is restricted to normative inferences in which there is only one type of obligation, no iteration of deontic operator and no mixed formulas. This is both a strength and a weakness of our approach. It is a strength insofar as the paradoxes which use these properties cannot be formulated in our framework. However, it is a weakness since we cannot formulate conditional obligations, which are of central importance in legal reasoning. The main contribution of this paper is that our method covers a portion of the intuitive validity of legal inferences which was not covered by other monadic frameworks in the literature. For instance, the principle of deontic consequences is not valid in standard deontic logic. Although the principle of deontic consequences can be formulated (and validated) in multimodal deontic logics which include a minimal necessity operator $\Box$, our aim was to develop a system which can easily be applied to test the validity of some basic legal inferences, which can be done with the help of graphical representations of arguments. Moreover, our intention was to introduce an alternative to the modal logic K, which is usually used as a building block for the construction of deontic logics. It is our view that our framework is better suited than K to formalize the transmission of the property of 'legally obligatory' between actions.

For future research, we intend to extend this method to arguments in which there are different types of obligations which can be iterated and are linked by a relation of hierarchy to prevent conflicts of obligations. We will also work on the representation of mixed formulas and conditional obligations. Another avenue will be to explore how this system can incorporate (or be incorporated in) the frameworks of dynamic and *stit* logics.

Bibliography

[1] C. Alchourrón and E. Bulygin. The expressive conception of norms. In R. Hilpinen, editor, *New Studies in Deontic Logic*, pages 95–124. D. Reidel Publishing Company, Dordrecht, 1981.

[2] C. Alchourrón. Logic without truth. *Ratio juris*, 3(1):46–67, 1990.

[3] A. J. J. Anglberger. Alternative Reductions for Dynamic Deontic Logics. In A. Hieke and H. Leitgeb, editors, *Reduction-Abstraction-Analysis*, volume 11, pages 179–191. Ontos Verlag, Frankfürt, 2009.

[4] L. Åqvist. Deontic logic. In D. M. G. e. F. Guenthner, editor, *Handbook of philosophical logic*, volume 8, pages 605–714. Kluwer Academic Publishers, Boston, 2e edition, 2001.

[5] G. Boella and L. van der Torre. A logical architecture of a normative system. In L. Goble and J.-J. C. Meyer, editors, *Deontic Logic and Artificial Normative Systems, 8th International Workshop on Deontic Logic in Computer Science, DEON 2006, Utrecht, The Netherlands, July 12-14, 2006, Proceedings*, volume 4048 of *Lecture Notes in Computer Science*, pages 24–35. Springer, 2006.

[6] J. Broersen. Action Negation and Alternative Reductions for Dynamic Deontic Logics. *Journal of Applied Logic*, pages 153–168, 2004.

[7] J. Carmo and A. J. Jones. Deontic Logic and Contrary-to-Duties. In D. M. G. e. F. Guenthner, editor, *Handbook of philosophical logic*, volume 8, pages 265–343. Kluwer Academic Publishers, Boston, 2e edition, 2001.

[8] H.-N. Castañeda. Acts, the logic of obligation, and deontic calculi. *Philosophical Studies*, 19:13–26, 1968.

[9] H.-N. Castañeda. The paradoxes of deontic logic. In R. Hilpinen, editor, *New Studies in Deontic Logic*, pages 37–85. D. Reidel Publishing Company, Dordrecht, 1981.

[10] B. F. Chellas. Conditional obligation. In S. Stenlund, editor, *Logical theory and semantic analysis*, pages 23–33. D. Reidel Publishing Company, Dordrecht, 1974.

[11] P.-A. Côté. *Interprétation des lois*. Les Éditions Thémis, Montréal, 3e edition, 2006.

[12] R. Demolombe. Obligations with deadlines: a formalization in dynamic deontic logic. *Journal of Logic and Computation*, 2012.

[13] J. Garson. *Modal logic for philosophers*. Cambridge University Press, Cambridge, 2006.

[14] J. Horty. *Agency and Deontic Logic*. Oxford University Press, 2001.

[15] J. Hughes and L. Royakkers. Don't ever do that!: Long-term duties in PD_eL. *Studia logica*, 89(1):59, 2008.

[16] A. J. I. Jones and I. Pörn. Ideality, sub-ideality and deontic logic. *Synthese*, 65:275–290, 1985.

[17] A. J. I. Jones and M. J. Sergot. A formal characterisation of institutionalised power. *Logic Journal of the IGPL*, 4(3):427–443, 1996.

[18] J. Jørgensen. Imperatives and logic. *Erkenntnis*, 7(1):288–296, 1937.
[19] D. Makinson and L. van der Torre. Input/Output Logics. *Journal of Philosophical Logic*, 29(4):383–408, 2000.
[20] J.-J. C. Meyer. A simple solution to the "deepest" paradox in deontic logic. *Logique et Analyse*, 117-118:81–90, 1987.
[21] J.-J. C. Meyer. A different approach to Deontic Logic: Deontic Logic Viewed as a Variant of Dynamic Logic. *Notre Dame Journal of Formal Logic*, 29:109–136, 1988.
[22] D. Nute, editor. *Defeasible deontic logic*. Kluwer Academic Publishers, Dordrecht, 1997.
[23] C. Peterson. *La logique déontique: Une application de la logique à l'éthique et au discours juridique*. Mémoire de maîtrise, Université de Montréal, 2011.
[24] C. Prisacariu and G. Schneider. A dynamic deontic logic for complex contracts. *Journal of Logic and Algebraic Programming*, 81(4):458–490, 2012.
[25] L. Royakkers. Extending Deontic Logic for the Formalization of Legal Rules. In *Law and Philosophy Library 36*. Dordrecht: Kluwer Academic Publishers, 1998.
[26] K. Segerberg. A deontic logic of action. *Studia logica*, 41:269–282, 1982.
[27] K. Segerberg. Blueprint for a dynamic deontic logic. *Journal of Applied Logic*, (7):388–402, 2009.
[28] K. Segerberg. DΔL: a dynamic deontic logic. *Synthese*, 185(S1):1–17, 2012.
[29] K. Solt. Deontic Alternative Worlds and the Truth-Value of 'OA'. *Logique et analyse*, 27:349–351, 1984.
[30] J. E. Tomberlin. Contrary-to-duty imperatives and conditional obligations. *Noûs*, 15:357–376, 1981.
[31] J. van Eck. A system of temporally relative modal and deontic predicate logic and it's philosophical applications. *Logique et analyse*, 25:249–290, 1982.
[32] J. van Eck. A system of temporally relative modal and deontic predicate logic and it's philosophical applications (2). *Logique et analyse*, 25:339–381, 1982.
[33] B. C. van Fraassen. The logic of conditional obligation. *Journal of Philosophical Logic*, 1:417–438, 1972.
[34] G. H. von Wright. Deontic logic. *Mind*, 60:1–15, 1951.
[35] G. H. von Wright. Deontic logics. *American philosophical quarterly*, 4:136–143, 1967.
[36] O. Weinberger. A philosophical approach to norm logic. *Ratio juris*, 14(1):130–141, 2001.
[37] J. Wolenski. Deontic logic and possible world semantics: A historical sketch. *Studia logica*, pages 273–282, 1990.

Appendix I

5 Completeness

The following lemmas will be useful. As a notational convention, we will write s^α_A to refer to the action Γ_A (i.e., the action described by the descriptive proposition A), and which is the m^{th} member of s^α. We also write $s^\alpha_{\neg A}$ instead of $\overline{s}^\alpha_A$.

Lemma 1. *If s^α satisfies O_iA, then $s^\alpha_A \in \mathcal{O}$.*

Proof. We proceed inductively on the length of the formula. Suppose that s^α satisfies O_iA but that $s^\alpha_A \notin O$. (The inductive step (HI) is that if the property holds for $l = n$, then it also holds for $l = n + 1$.)

1. *A is p_i, thus s^α satisfies p_i and $s^\alpha_{p_i} \in \mathcal{O}$.*
2. *A is $\neg B$, thus $\overline{s}^\alpha$ satisfies B.*
   (a) *B is p_i, thus $\overline{s}^\alpha$ satisfies p_i and $s^\alpha_{\neg p_i} \in \mathcal{O}$.*
   (b) *B is $\neg C$, thus $\overline{s}^\alpha$ satisfies $\neg C$, meaning that s^α satisfies C and by (HI) $s^\alpha_C \in \mathcal{O}$, and thus $s^\alpha_{\neg\neg C} \in \mathcal{O}$ by C2 since $\models_{PL} C \supset \neg\neg C$ and $\Gamma_C \sqsubseteq \Gamma_{\neg\neg C}$.*
   (c) *B is $C \supset D$, thus $\overline{s}^\alpha$ satisfies $C \supset D$, meaning that s^α satisfies C and $\overline{s}^\alpha$ satisfies D. But by (HI) $s^\alpha_C \in \mathcal{O}$ and $s^\alpha_{\neg D} \in \mathcal{O}$, and since $\models_{PL} C \supset (\neg D \supset \neg(C \supset D))$, we have $s^\alpha_C \sqsubseteq s^\alpha_{\neg D \supset \neg(C \supset D)}$, thus by C3 we have $s^\alpha_{\neg D} \sqsubseteq s^\alpha_{\neg(C \supset D)}$, and by C2 we obtain $s^\alpha_{\neg(C \supset D)} \in \mathcal{O}$ since $s^\alpha_{\neg D} \in \mathcal{O}$.*
3. *A is $B \supset C$, thus either*
   (a) *s^α satisfies $\Gamma_m = B \supset C$ and thus $s^\alpha_{B \supset C} \in \mathcal{O}$*
   (b) *s^α satisfies C ($\Gamma_B \neq \Gamma_C$) and by (HI) $s^\alpha_C \in \mathcal{O}$ and by C2 $s^\alpha_{B \supset C} \in \mathcal{O}$ since $\models_{PL} C \supset (B \supset C)$ and thus $\Gamma_C \sqsubseteq \Gamma_{B \supset C}$.*

Lemma 2. *If $s^\alpha_A \in \mathcal{O}$, then s^α satisfies O_iA.*

Proof. We proceed inductively on the length of the formula. Suppose that $s^\alpha_A \in \mathcal{O}$ but that s^α does not satisfy O_iA.

1. *A is p_i, thus $s^\alpha_{p_i} \in \mathcal{O}$ and so s^α satisfies p_i.*
2. *A is $\neg B$, thus $\overline{s}^\alpha$ does not satisfy B.*
   (a) *B is p_i, thus $s^\alpha_{\neg p_i} \in \mathcal{O}$ and so $\overline{s}^\alpha$ satisfies p_i*
   (b) *B is $\neg C$, thus $\overline{s}^\alpha$ does not satisfy $\neg C$, meaning that s^α does not satisfy C. Since $\models_{PL} \neg\neg C \supset C$, we obtain $s^\alpha_{\neg\neg C} \sqsubseteq s^\alpha_C$ and by C2 we have $s^\alpha_C \in \mathcal{O}$, and by (HI) s^α satisfies C.*
   (c) *B is $C \supset D$, thus $\overline{s}^\alpha$ does not satisfy $C \supset D$, meaning that either s^α does not satisfy C or $\overline{s}^\alpha$ does not satisfy D. By hypothesis, we have $s^\alpha_{\neg(C \supset D)} \in \mathcal{O}$, and by PL we obtain $s^\alpha_{\neg(C \supset D)} \sqsubseteq s^\alpha_C$ and $s^\alpha_{\neg(C \supset D)} \sqsubseteq s^\alpha_{\neg D}$ since $\models_{PL} \neg(C \supset D) \supset C$ and $\models_{PL} \neg(C \supset D) \supset \neg D$. Therefore, by C2 we obtain $s^\alpha_C \in \mathcal{O}$ and $s^\alpha_{\neg D} \in \mathcal{O}$, which by (HI) implies that s^α satisfies C and $\overline{s}^\alpha$ satisfies D.*

3. *A is $B \supset C$, thus s^α does not satisfy $\Gamma_m = B \supset C$ and either s^α does not satisfy C or $s^\alpha_B = s^\alpha_C$. However, by definition if s^α does not satisfy Γ_m, it implies that $s^\alpha_m \notin O$, that is $s^\alpha_{B \supset C} \notin O$.*

Lemma 3. *If $\Gamma_A \sqsubseteq \Gamma_B$, $\not\models_{PL} B$ and s^α satisfies O_iA, then s^α satisfies O_iB.*

Proof. Assume $\Gamma_A \sqsubseteq \Gamma_B$, $\not\models_{PL} B$ and s^α satisfies O_iA but s^α does not satisfy O_iB. By lemma 1 we have $s^\alpha_A \in \mathcal{O}$, thus by C2 $s^\alpha_B \in \mathcal{O}$ and by lemma 2 s^α satisfies O_iB.

Lemma 4. *(A1) preserves validity.*

Proof. Suppose that (A1) does not preserve validity. Thus, we have $\vdash_{OL} \neg(O_iA \wedge O_i\neg A)$ and $\not\models_{\mathcal{N}} \neg(O_iA \wedge O_i\neg A)$. However, if $\not\models_{\mathcal{N}} \neg(O_iA \wedge O_i\neg A)$, then $\models_{\mathcal{N}} O_iA \wedge O_i\neg A$. It follows that s^α satisfies O_iA and $O_i\neg A$, meaning that s^α satisfies A and it satisfies $\neg A$. By lemma 1, it implies that both $s^\alpha_A \in \mathcal{O}$ and $s^\alpha_{\neg A} \in \mathcal{O}$, which contradicts C1.

Lemma 5. *(R1) preserves validity.*

Proof. Assume that (R1) does not preserve validity. This means that we have a situation where $O_iA \vdash_{OL} O_iB$ is obtained by the use of (R1) but $O_iA \not\models_{\mathcal{N}} O_iB$, that is $\models_{\mathcal{N}} O_iA$ and $\not\models_{\mathcal{N}} O_iB$, and so s^α satisfies O_iA but does not satisfy O_iB. By the use of (R1), we know that $A \supset B$ is true by hypothesis and that $\not\models_{PL} B$, and moreover that $\Gamma_A \sqsubseteq \Gamma_B$. Therefore, by lemma 3 s^α satisfies O_iB.

Theorem 1 (Adequacy). *If $\vdash_{OL} A$, then $\models A$.*

Proof. Since OL is based upon PL, it suffices to show that (A1) and (R1) preserve validity, which follows from lemmas 4 and 5.

Lemma 6 (Lindenbaum's lemma). *OL has a maximally consistent extension.*

Suppose $K = \bigcup_0^\infty K_i$, with $K_0 = OL$ the smallest set of wffs of $\mathcal{L}$ closed under the rules of PL, (A1) and (R1). Let $A_1, \ldots, A_n, \ldots$ be an arbitrary enumeration of OL's wffs. If $\vdash_{K_{n-1}} \neg A_n$, then $K_n = K_{n-1}$, else $K_n = K_{n-1} \cup \{A_n\}$. This way, we have K_i extension of OL for all $i \geq 0$. It is obvious that K is maximal since by construction either $A_i \in K$ or $\neg A_i \in K$ for all i. We now show that K is consistent.

Proof. Assume $K \vdash \bot$. Thus, there is a finite proof of $\bot$ from a finite subset of K, meaning that there is K_n such that $K_n \subset K$ and $K_n \vdash \bot$. If K_n is inconsistent, it implies that by construction the proposition A_n added to K_{n-1} broke K_n's consistency. However, such a situation is impossible. Indeed, this means that $K_n \vdash A_n \wedge \neg A_n$, with $K_n = K_{n-1} \cup \{A_n\}$ and $K_{n-1} \vdash \neg A_n$. But if $K_{n-1} \vdash \neg A_n$, then $K_n = K_{n-1}$, thus $K_n \nvdash \bot$. And if $K_{n-1} \nvdash \neg A_n$, then $K_n = K_{n-1} \cup \{A_n\}$, thus $K_n \nvdash \bot$ since $K_n \nvdash \neg A_n$. Therefore, K is consistent.

Lemma 7. *$\mathcal{N}$ is a model of K.*

Proof. Assume that $\mathcal{N}$ is not a model of some maximally consistent extension K. It follows that there is a proposition A such that $\vdash_K A$ and $K \not\models_{\mathcal{N}} A$. But if $\vdash_K A$, it means that there is a finite proof of A from a finite subset of K, thus $\vdash_{K_n} A$ and $K_n \not\models_{\mathcal{N}} A$. However, since every subset of K is an extension of OL, it implies that $K_n \vdash_{OL} A$. By theorem 1, if $K_n \vdash_{OL} A$, then $K_n \models_{\mathcal{N}} A$, which contradicts our first hypothesis. Therefore, $\mathcal{N}$ is a model of K.

Theorem 2 (Completeness). *If $H \models C$, then $H \vdash_{OL} C$.*

Proof. Assume that there is a maximally consistent extension of OL where $H \models_{\mathcal{N}} C$ and $H \nvdash_K C$. Since K is maximally consistent, it follows that $H \vdash_K \neg C$, and since $\mathcal{N}$ is a model of K it implies that $H \models_{\mathcal{N}} \neg C$. However, if $H \models_{\mathcal{N}} C$, then $H \not\models_{\mathcal{N}} \neg C$, which contradicts our first hypothesis. Therefore, if $H \models_{\mathcal{N}} C$, then $H \vdash_K C$ for any $\mathcal{N}$.

Appendix II

6 Validity of legal reasoning

6.1 Graphical representation

The idea that lies behind this approach is that, according to the semantical dichotomy, a scenario w is divided in two parts : one descriptive ($\mathfrak{D}$) and the other normative ($\mathfrak{N}$), which contains a set of obligations $\mathcal{O}$. A scenario w is inconsistent if either $\mathfrak{D}$ or $\mathfrak{N}$ is. An argument's validity can be tested through a counterexample: assume a scenario w in which the premises are true but the conclusion is false and see if it is inconsistent. If it is, then the argument is valid. If it is not inconsistent (i.e., it is consistent), then it is invalid, meaning that there is a truth value assignment where the premises are true but the conclusion is false. The propositional rules (figure 1) representing schematically truth conditions for complex propositions are quite straightforward [cf. 13, p.91].

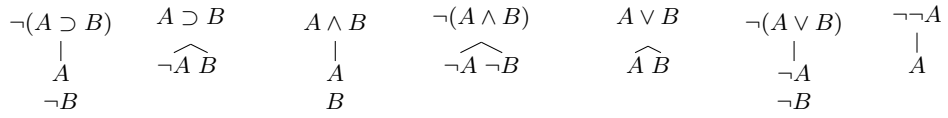


Fig. 1. Propositional rules

The semantical dichotomy implies that normative and descriptive propositions are not true in the same conditions. While the truth of a descriptive proposition can be represented by the fact that it belongs to the descriptive part of a scenario, the truth of a normative proposition depends upon a norm established by some authority: A is obligatory if and only if there is a norm which makes the action described by A (i.e., Γ_A) an obligation. From a propositional standpoint, complex normative formulas behaves schematically as complex descriptive formulas since both are composed of the same logical connectives (figure 1). The difference of truth conditions between normative and descriptive propositions can be seen through the representation of normative atoms. If $O_i A$ is true for w, then

1. there is a norm (established by authority i) which makes Γ_A obligatory;
2. the action described by A pertains to a set of obligations.

Since logical connectives can (classically) be reduced to composition of $\neg$ and $\supset$, we will only represent schematically truth conditions for $O_i p$, $O_i \neg A$ and $O_i(A \supset B)$ (figure 2). These rules, combined with conditions C1, C2 and C3, allow us to verify the validity of a normative inference.

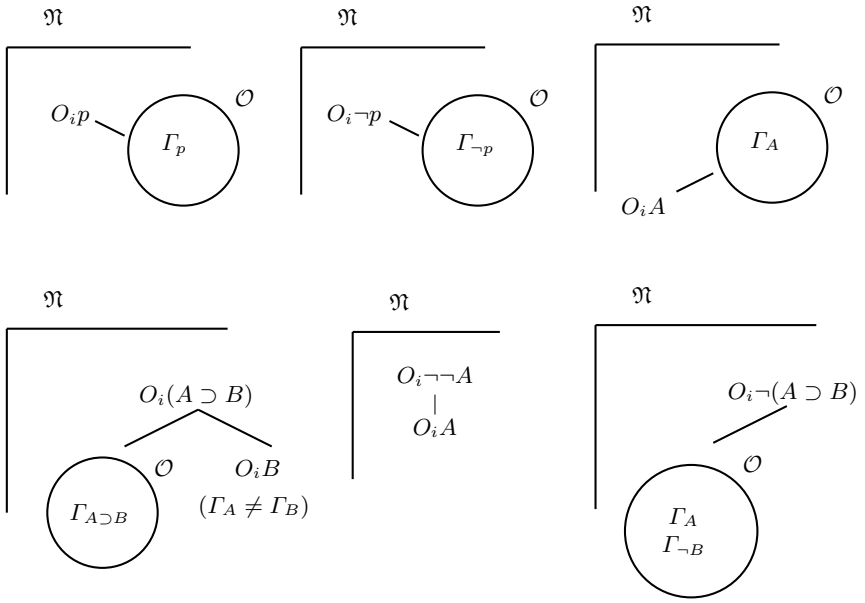


Fig. 2. Truth conditions of normative atoms

As an example, let us test (R1)'s validity with $n = 1$ (figure 3). Assume that w is a counter-example for (R1). Then, place the assumptions in their respective descriptive or normative part of w. According to the rules for normative atoms, Γ_p is in $\mathcal{O}$. Since $p \supset q$ is assumed to be true in the descriptive part of w, it follows that $\Gamma_p \sqsubseteq \Gamma_q$ holds in its normative part. By C2 we obtain Γ_q in O, meaning that we can conclude Oq in the normative part of w. The only branch in the normative part closes (schematically represented by $\times$) since it contains a normative contradiction (i.e., $O_i q$ and $\neg O_i q$). Therefore, there is no possible scenario in which the premises of (R1) are true while its conclusion is false, hence the proof of (R1)'s validity. The proof can be rephrased as following. Assume that the action described by p is obligatory, that p implies q but that the action described by q is not obligatory. Since the action described by p is obligatory, it follows that it pertains to a set of obligation. Considering that p implies q is taken to be true in the descriptive part of w, it follows that there is a semantical relation between the action described by p and the action described by q. From this relation and the condition C2, we can conclude that the action described by q also pertains to the set of obligations, and thus the action described by q is obligatory, which contradicts our hypothesis. (Ask author for more examples.)

We now list some semantical properties of the model. These properties are useful for both the formal semantical proofs and the graphical representation. (Ask author for proofs. Most of them are straightforward from C1, C2, C3 and PL).

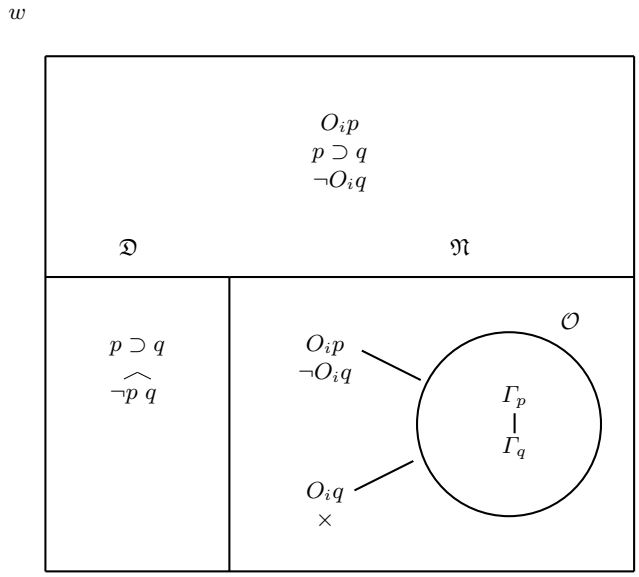


Fig. 3. Test of (R1)'s validity

Lemma 8. *If $\Gamma_A \in \mathcal{O}$, then $\Gamma_{B \supset A} \in \mathcal{O}$ for any B.*

Lemma 9. *If $\Gamma_A \in \mathcal{O}$ and $\Gamma_B \in \mathcal{O}$, then $\Gamma_{A \wedge B} \in \mathcal{O}$.*

Lemma 10. *If $\Gamma_{A \wedge B} \in \mathcal{O}$, then $\Gamma_A \in \mathcal{O}$.*

Lemma 11. *If $\Gamma_A \in \mathcal{O}$ and $\Gamma_{A \supset B} \in \mathcal{O}$, then $\Gamma_B \in \mathcal{O}$.*

Lemma 12 (De Morgan). *$\Gamma_{A \wedge B} \in \mathcal{O}$ if and only if $\Gamma_{\neg(\neg A \vee \neg B)} \in \mathcal{O}$.*

Lemma 13. *The next lemmas are consequences of (De Morgan) and lemma 8.*

$$\text{If } \Gamma_A \in \mathcal{O}, \text{ then } \Gamma_{A \vee B} \in \mathcal{O} \tag{6}$$
$$\text{If } \Gamma_A \in \mathcal{O}, \text{ then } \Gamma_{\neg(\neg A \wedge \neg B)} \in \mathcal{O} \tag{7}$$
$$\text{If } \Gamma_{\neg A} \in \mathcal{O}, \text{ then } \Gamma_{\neg A \vee \neg B} \in \mathcal{O} \tag{8}$$
$$\text{If } \Gamma_{\neg A} \in \mathcal{O}, \text{ then } \Gamma_{\neg(A \wedge B)} \in \mathcal{O} \tag{9}$$
$$\text{If } \Gamma_{\neg(A \vee B)} \in \mathcal{O}, \text{ then } \Gamma_{\neg A} \in \mathcal{O} \tag{10}$$
$$\text{If } \Gamma_{\neg(\neg A \vee \neg B)} \in \mathcal{O}, \text{ then } \Gamma_A \in \mathcal{O} \tag{11}$$

6.2 Soundness of legal inference

In addition to the formal method developed so far, one might want some hints to test the soundness of a legal inference. Without pretending to be exhaustive,

we only give some philosophical insights regarding relevant questions one might try to answer. A sound inference is valid and has true (or acceptable) premises. The second question (after the validity test) is thus are the premises true? The truth of a normative proposition depends upon the fact that there is an authority which establishes that norm. Hence, one must first contextualize the normative proposition to a legal authority: is it true according to the Constitution? the Civil Code? the Criminal Code? the Canadian Charter of Rights and Freedom? It is noteworthy that some inferences can be fallacious when the meaning of 'ought' is not the same throughout the argument.

The context of the argument is also relevant. Consider the premise 'Paul ought to tell the truth'. Assume that 'ought' is understood legally. While one is not legally obliged to tell the truth in one's day to day life, one *is* however legally obliged to tell the truth in a court of justice. It might also be useful to analyze the descriptive propositions used in conjunction with (R1). The conditional must be analyzed in terms of necessity and sufficiency. Is the conditional representing a semantical relation of entailment between two actions? Is the action described by the consequent necessarily entailed by the action described by the antecedent?

Furthermore, one must contextualize the descriptive propositions according to the normative authority. For instance, the meaning of the proposition 'Paul respects his neighbor' will vary according to the legal authority. Indeed, the meaning of 'Paul must respect his neighbor' will vary depending whether the legal authority is the Civil Code or the Criminal Code (or the Canadian Charter of Rights and Freedom). When this is done, the question that must be answered is what legally counts as a lack of respect from the Civil Code's point of view? In other words, one must answer the question 'what action counts as A from the authority's standpoint?'.[13]

Now, consider the following argument and assume a context in which premises P_2 and P_3 are acceptable. Assume that P_1 'Paul ought to rescue Peter from drowning', P_2 'If Paul rescues Peter, then Paul calls 911' and P_3 'If Paul calls 911, then Paul breaks into Sam's house'. Therefore, we can conclude that 'Paul ought to break into Sam's house'. Assuming that breaking and entering is legally forbidden, this argument leads to a conflict of obligations, hence to a contradiction into our current framework. The contradiction can be obtained with the addition of the premise P_0 ' Paul ought to not break into Sam's house'. But although valid, this argument is not sound: either P_0 or P_1 must be rejected. The conflict of obligations can be overruled with the help of the relation of hierarchy since it allows one to prioritize P_1 over P_0. It is not only likely that breaking into Sam's house to call 911 and save Peter from drowning would not count as breaking and entering per se, but it is also likely that preserving the integrity of one's life would overrule preserving the integrity of one's property. Hence, premise P_0 can be rejected and the conflict of obligations is solved.

A final clue is to consider the principle *ought implies can*. Replace P_2 and P_3 in the aforementioned example by P_4 'If Paul rescues Peter from drowning, then Paul jumps in the water to retrieve Peter'. From P_1 and P_4 one can conclude

[13] For an analysis of 'count as', see [17] and [5].

that ‘Paul ought to jump in the water to retrieve Peter’. However, if Paul cannot swim (e.g. if Paul has two broken arms) then the action described by ‘rescuing Peter from drowning’ does not entail the action described by ‘jumping in the water to retrieve Peter’. Thus, in this context, one could reject premise P_2 on the ground that it is a violation of the ought imply can principle.

A Memetic Algorithm for Solving the Generalized Vehicle Routing Problem

P.C. Pop[1], O. Matei[2]

[1] Dept. of Mathematics and Informatics, Technical Univ. of Cluj-Napoca, Romania
petrica.pop@ubm.ro
[2] Dept. of Electrical Engineering, Technical University of Cluj-Napoca, Romania
oliviu.matei@holisun.com

Abstract. The generalized vehicle routing problem (GVRP) is one of the challenging combinatorial optimization problems that finds a lot of practical applications especially in the field of distribution, collection and logistics. The GVRP is a natural extension of the classical vehicle routing problem (VRP), obtained by replacing the nodes of a graph with sets of nodes (clusters) and looking for the minimum-cost collection of routes subject to capacity restrictions, from a given depot to a number of predefined clusters passing through exactly one node from each cluster. This paper describes an efficient memetic algorithm (MA) for solving the GVRP obtained by combining a genetic algorithm (GA) with a powerful local search (LS) procedure. The preliminary experiments on benchmark instances show that the proposed algorithm compares favorably to all previous methods.

Keywords: generalized vehicle routing problem, memetic algorithms, genetic algorithms, local search.

1 Introduction

The *generalized vehicle routing problem* (GVRP) is a natural extension of the classical vehicle routing problem (VRP) and it was introduced by Ghiani and Improta [3]. The GVRP consists in finding the minimum-cost delivery or collection of routes, subject to capacity restrictions, from a given depot to a number of predefined clusters passing through one node from each cluster. The problem can be viewed as a particular type of location-routing problem.

The GVRP has several real world applications mainly in network design. Some applications of the GTSP can be extended naturally to GVRP and several other situations can be modeled as a GVRP: the post-box collection problem, routing vessels in maritime transportation, health care logistics, the design of tandem configurations for automated guided vehicles, urban waste collection problem, survivable telecommunication network design, etc. For a more detailed description of the GVRP applications we refer to [1, 2].

The existing literature on the GVRP is rather scarce: Ghiani and Improta [3] showed that the problem can be transformed into a capacitated arc routing

problem (CARP) and Baldacci et al. [1] proved that the reverse transformation is valid. Integer programming formulations for the GVRP have been proposed by Bektas *et al.* [2] and Pop *et al.* [7].

The difficulty of obtaining optimal solutions for the GVRP has led to the development of heuristics and metaheuristics: the first such algorithms were the ant colony algorithm of Pop *et al.* [9], the genetic algorithm of Pop *et al.* [8] and more recently a branch-and-cut algorithm and an adaptive large neighborhood search proposed by Bektas *et al.* [2] and an incremental tabu search heuristic described by Moccia *et al.* [5].

The aim of this paper is to develop an efficient memetic algorithm (MA), obtained by combining a genetic algorithm with a powerful local search procedure, for solving the GVRP.

The remainder of the paper is organized as follows. Section 2 formally states the GVRP and introduces the corresponding notations. Section 3 recalls the MA methodology and describes how the GVRP based on the MA framework can be solved. Section 4 presents and analyses the results of the computational experiments. The obtained results are compared with the state-of-the-art algorithms for solving the GVRP: the branch-and-cut algorithm [2] and the tabu search (TS) heuristic [5]. Finally, the last section concludes the paper.

2 Problem description

Formally, the GVRP is defined on a directed graph $G = (V, A)$ with $V = \{0, 1, 2, ..., n\}$ as the set of nodes and the set of arcs A. The set of nodes consists of vertex $v = 0$ which represents the depot and the vertices $v = 1, ..., n$ which represent the customers and it is partitioned into $k+1$ mutually exclusive nonempty subsets, called clusters, $V_0, V_1, ..., V_k$ (i.e. $V = V_0 \cup V_1 \cup ... \cup V_k$ and $V_l \cap V_p = \emptyset$ for all $l, p \in \{0, 1, ..., k\}$ and $l \neq p$). The cluster V_0 is a singleton consisting of the depot vertex $v = 0$. The set of arcs is defined by arcs connecting vertices belonging to different clusters called intra-cluster arcs. Inter-cluster arcs do not exist. A distance d_{ij} is associated to each arc $(i, j) \in A$ and it represents the travel cost between vertices i and j. Each customer has a certain amount of demand and the total demand of each cluster can be satisfied via any of its nodes. There exists m identical vehicles, each with a given capacity Q.

The generalized vehicle routing problem (GVRP) consists in finding the minimum total cost collection of routes starting and ending at the depot, such that each cluster should be visited exactly once, the entering and leaving nodes of each cluster is the same and the sum of all the demands of any route does not exceed the capacity of the vehicle Q.

Therefore the GVRP involves the following two related decisions:

- choosing a node subset $S \subseteq V$, such that $|S \cap V_i| = 1$, for all $i = 1, ..., k$.
- finding a minimum cost collection of routes in the subgraph of G induced by S, fulfilling the capacity constraints.

We will call such a route (i.e. a route visiting exactly one node from a number of clusters and fulfilling the capacity constraints) a *generalized route*. An illustrative scheme of the GVRP and a feasible collection of routes is shown in the next figure.

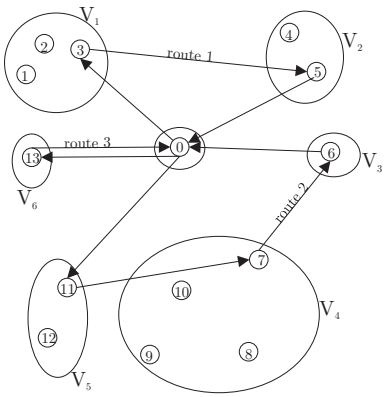


Figure 1: An example of a feasible solution of the generalized vehicle routing problem (GVRP)

The GVRP reduces to the classical Vehicle Routing Problem (VRP) when all the clusters are singletons and to the Generalized Traveling Salesman Problem (GTSP) when $Q = \infty$.

Concerning the complexity, the GVRP is an NP-hard optimization problem because it includes the generalized traveling salesman problem as a special case.

3 A memetic algorithm for solving the GVRP

Memetic algorithms have been introduced by Mascato [6] to denote a family of metaheuristic algorithms that emphasis on the used of a population-based approach with separate individual learning or local improvement procedures for problem search. Therefore a memetic algorithm is a genetic algorithm (GA) hybridized with a local search procedure to intensify the search space.

Genetic algorithms are not well suited for fine-tuning structures which are close to optimal solutions. Therefore, incorporating of local improvement operators into the recombination step of a GA is essential in order to obtain a competitive GA.

Our effective heuristic algorithm for solving the GVRP is a memetic algorithm, which combines the power of genetic algorithm with that of local search. The general scheme of our heuristic is:

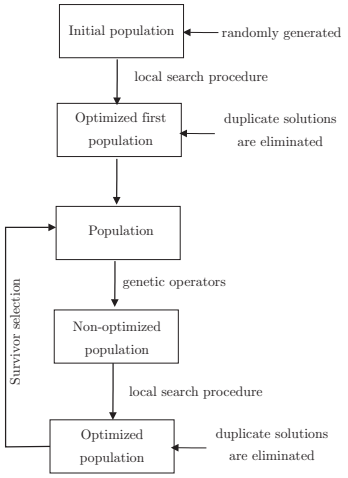


Figure 2: Generic form of our memetic algorithm

3.1 The genetic algorithm

We used a natural, compact and efficient encoding of solutions of GVRP similar to that described by Pop *et al.* [8]. Specifically, 0 represents the depot and each customer is tagged with a non-duplicated natural number from 1 to n. We represent a chromosome by a variable length array so that the gene values correspond to the nodes selected to form the collection of routes which are delimited by 0 representing the depot.

The corresponding chromosome representation of the feasible solution of the GVRP presented in figure 1 is: (3 5 0 6 7 11 0 13), where the values $\{1, ..., 13\}$ represent the customers while the depot denoted by 0 is the route splitter. Route 1 begins at the depot then visits customers 3 and 5 belonging to the clusters V_1, respectively V_2 and returns to the depot. Route 2 starts at the depot and visits the customers 6-7-11 belonging to the clusters $V_3 - V_4 - V_5$. Finally, in route 3 only customer 13 from the cluster V_6 is visited.

The first step of any genetic algorithm is to generate a set of possible solutions as an initial generation or population to the problem. Although it seems simple, the convergence, the performance and the ability of the GA are critically affected by the initial generation.

In the case of the GVRP we carried out experiments with the initial population generated randomly and with an initial population of structured solutions. The initial population generated randomly has the advantage that is representative from any area of the search space. In order to generate the population of structured solutions we used a Monte Carlo based method. Based on computational experiments we observed that this method assures an initial population with a average fitness with 20% better than a population generated entirely randomly, but the major drawback was that such a population lacks the diversity needed to obtain near-optimal solutions.

Analyzing the advantages and disadvantages of these ways to generate the initial population and based on computational experiments we decided to choose the initial population randomly.

In order for genetic algorithms to work effectively, it is necessary to be able to evaluate how "good" a potential solution is relative to other potential solutions. In our case, the fitness value of a feasible solution, i.e. collection of routes, is given by the sum of the costs of the arcs selected in the routes. The aim is to find the minimum cost collection of routes.

The crossover operator requires some strategy to select two parents from previous generation. We used in our algorithm an elitist approach: the parents are chosen randomly between the best 40% (parameter chosen based on preliminary experiments) of all the solutions in the previous generation. We select randomly two chromosomes from the best 40% of the all the solutions of the previous generation, assign the cheapest (w.r.t. cost minimization) to parent 1 and repeat the procedure to select parent 2.

Offspring are produced from two parents using the following 2-point order standard crossover procedure: it creates offspring which preserve the order and position of symbols in a subsequence of one parent while preserving the relative order of the remaining symbols from the other parent. It is implemented by selecting two random cut points which define the boundaries for a series of copying operations. First, the symbols between the cut points are copied from the first parent into the offspring. Then, starting just after the second cut-point, the symbols are copied from the second parent into the offspring, omitting any symbols that were copied from the first parent. When the end of the second parent sequence is reached, this process continues with the first symbol of the second parent until all the symbols have been copied into the offspring. The second offspring is produced by swapping round the parents and then using the same procedure.

Next we present the application of the proposed 2-point order crossover in the case of the problem presented in fig. 1. We assume two well-structured parents chosen randomly, with the cutting points between nodes 2 and 3, respectively 5 and 6:

$$P_1 = (13\ 0 \mid 3\ 5\ 0 \mid 11\ 7\ 6) \quad P_2 = (4\ 2 \mid 13\ 0\ 11 \mid 10\ 6)$$

Note that the length of each individual differs according to the number of routes. According to figure 1, the cluster representation of the parents is as follows:

$$C_1 = (6\ 0 \mid 1\ 2\ 0 \mid 5\ 4\ 3) \quad C_2 = (2\ 1 \mid 6\ 0\ 5 \mid 4\ 3)$$

The sequences between the two cutting-points are copied into the two offspring:

$$O_1 = (x\ x \mid 3\ 5\ 0 \mid x\ x\ x) \quad O_2 = (x\ x \mid 13\ 0\ 11 \mid x\ x)$$

The nodes of the parent P_1 are copied into the offspring O_2 if O_2 does not contain already nodes in the same clusters as the nodes of P_1. The sequence of the nodes of P_1 is $13 - 0 - 3 - 5 - 0 - 11 - 7 - 6$, and the clusters are

$6-0-1-2-0-5-4-3$. Note that the cluster 6 is already represented in O_2 by the node 13 and the cluster 5 by the node 11. The depot (0) is kept any way as it is a splitter. Therefore the remaining sequence of nodes in P_1 is $0-3-5-0-7-6$. Therefore the offspring O_2 is: $O_2 = (0\ 3 \mid 13\ 0\ 11 \mid 5\ 0\ 7\ 6)$.

Then the nodes of the parent P_2 are copied into the offspring O_1 in the same manner. The nodes of the clusters not present in O_1 are copied into the remaining positions: $O_1 = (13\ 0 \mid 3\ 5\ 0 \mid 11\ 10\ 6)$.

Mutation is a genetic operator that alters one ore more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible.

We use in our GA two random mutation operators: the first one (intra-route mutation) selects randomly a cluster to be modified and replaces its current node by another one randomly selected from the same cluster and the second one (inter-route mutation) is a swap operator, it picks two random locations in the solution vector and swaps their values. The new chromosome is accepted directly if it results in a feasible GVRP, otherwise the route that exceeds the vehicle capacity is decomposed into several ones as describe before.

The selection process is deterministic. In our algorithm we use the $(\mu + \lambda)$ selection, where μ parents produce λ offspring. The new population of $(\mu + \lambda)$ is reduced again to μ individuals by a selection based of the "survival of the fittest" principle. In other words, parents survive until they are suppressed by better offspring. It might be possible for very well adapted individuals to survive forever.

The genetic parameters are very important for the success of a GA, equally important as the other aspects, such as the representation of the individuals, the initial population and the genetic operators. The most important parameters are: the population size μ has been set to 2 times the number of the clusters, the intermediate population size λ was chosen twice the size of the population: $\lambda = 2 \cdot \mu$ and the mutation probability was set at 10%.

The loop of chromosome generations is terminated when certain conditions are met. When the termination criteria are met, the elite chromosome is returned as the best solution found so far. In our algorithm we used the termination condition based on the number of generations, namely 10^4 generations seems necessary to sufficiently explore the solution space.

3.2 Local improvement procedure

Classical GAs are not aggressive enough for some combinatorial optimization problems. One possibility to obtain more competitive heuristics is to combine the GAs with local search procedures.

For each solution belonging to the current generation we use a local improvement procedure that runs several local search heuristics sequentially. Once an improvement move is found, it is immediately executed. In our algorithm we used the following local search heuristics:

– *2-opt intra-route moves*: consist of eliminating any two non-adjacent edges in each of the routes of the solution and reconnect the two resulting paths in a different way to obtain a different route. An improvement is obtained if the cost of the new collection of routes is less than the cost of the initial collection of routes. The heuristic applies all the improvements found.
– *Cluster route optimization (CO)*: uses the shortest path algorithm to find the best node from each cluster when the order of visiting the clusters is given.
Given a collection of r global routes of form $(V_0, V_{k_1}, ..., V_{k_p})$ in which the clusters are visited, we show that the best feasible route R^* (w.r.t cost minimization), i.e. a collection of r generalized routes visiting the clusters according to the given sequence can be done in polynomial time, by solving the following r shortest path problems.
For each sequence of visiting the clusters $(V_0, V_{k_1}, ..., V_{k_p})$, the best generalized route visiting the clusters according to the given sequence can be determined in polynomial time by constructing a layered network (LN) with $p + 2$ layers corresponding to the clusters $V_0, V_{k_1}, ..., V_{k_p}$ and in addition we duplicate the cluster V_0. The layered network contains all the nodes of the clusters $V_0, V_{k_1}, ..., V_{k_p}$ plus an extra nodes $0' \in V_0$.

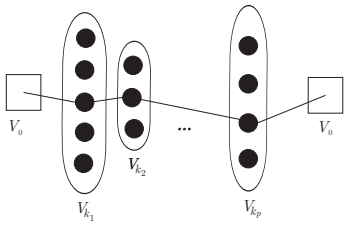


Fig. 3: Example showing a route visiting the clusters $V_0, V_{k_1}, ..., V_{k_p}$ in the constructed layered network LN

We consider paths from 0 to $0'$, $0' \in V_0$, that visits exactly one node from each cluster $V_{k_1}, ..., V_{k_p}$, hence it gives a feasible generalized route. Conversely, every generalized route visiting the clusters according to the sequence $(V_0, V_{k_1}, ..., V_{k_p})$ corresponds to a path in the layered network from $0 \in V_0$ to $0' \in V_0$.
Therefore, it follows that the best (w.r.t cost minimization) collection of routes R^* can be found by determining r shortest paths from $0 \in V_0$ to the corresponding $0' \in V_0$ with the property that visits exactly one node from each of the clusters $(V_{k_1}, ..., V_{k_p})$.
– *Neighborhood cross (NC)*: this neighborhood includes all feasible solutions obtained by the replacement of two arc by other two. Given two different routes r_1 and r_2, an arc is deleted from each of them dividing the routes into two components that are combined such that a new route is made of the first component of r_1 and the second of r_2 and another new route is made of the first component of r_2 and the second component of r_1.

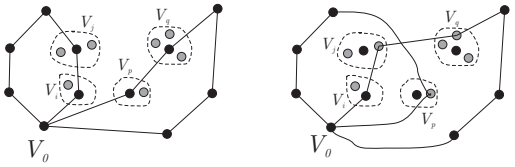


Fig. 4. An example of a possible string cross

– *Neighborhood exchange (NE)*: two strings of at most k vertices are exchanged between two routes. A new solution is obtained from the current one by selecting k vertices from one route and exchanging them with k vertices belonging to another route. The exchange is acceptable only if it produces a feasible solution.

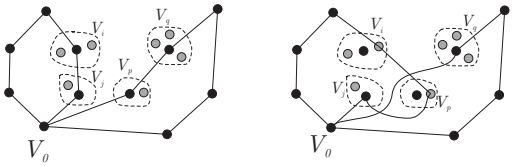


Figure 5: An example of a possible string exchange for $k = 1$

– *Neighborhood relocation (NR)*: a string of at most k vertices is moved from one route to another ($k = 1$ or $k = 2$). This neighborhood includes all feasible solutions obtained by deleting one or two vertices from their route r_1 and inserting them into a route r_2.

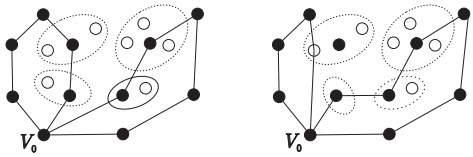


Figure 6: An example of a possible string relocation

Our improvement procedure applies all the described local search heuristics cyclically.

4 Computational results

In order to asses the performance of our proposed memetic algorithm for solving the GVRP, we conducted our experiments on a set of instances generated

through an adaptation of the existing instances in the CVRP-library available at *http://branchandcut.org/VRP/data/*. The naming of the generated instances follows the general convention of the CVRP instances available online, and follows the general format $X - nY - kZ - C\Omega - V\Phi$, where X corresponds to the type of the instance, Y refers to the number of vertices, Z corresponds to the number of vehicles in the original CVRP instance, Ω is the number of clusters and Φ is the number of vehicles in the GVRP instance. These instances were used by Bektas *et al.* [2] and Moccia *et al.* [5] in their computational experiments.

We considered for our instances as in [2, 5] two clustering procedures one with $\theta = 2$ and the other one with $\theta = 3$. However, the solution approach proposed in this paper is able to handle any cluster structure.

The testing machine was an Intel Dual-Core 1,6 GHz and 1 GB RAM. The operating system was Windows XP Professional. The algorithm was developed in Java, JDK 1.6.

In Tables 1 and 2, we summarize the results of all methods on a set of 30 test small to medium instances with $\theta = 2$ and $\theta = 3$. The first column in the tables give the name of the instances, the second column provides the values of the best lower bounds in the branch-and-cut tree [2]. Next three columns contains the values of the best solutions obtained using the adaptive large neighborhood search (ALNS) [2], the incremental tabu search (ITS) [5] and our memetic algorithm.

Table 1. Computational results on small and medium instances with $\theta = 2$

Instance	*LB* [2]	*ALNS* [2]	*ITS* [5]	*MA*
A-n32-k5-C16-V2	519	519	519	519
A-n34-k5-C17-V3	489	489	489	489
A-n38-k5-C19-V3	476	476	476	476
A-n45-k6-C23-V4	613	613	613	613
A-n63-k9-C32-V5	900.3	912	912	908
B-n31-k5-C16-V3	441	441	441	441
B-n34-k5-C17-V3	472	472	472	472
B-n38-k6-C19-V3	451	451	451	451
B-n45-k5-C23-V3	497	497	497	497
B-n63-k10-C32-V5	816	816	816	816
P-n16-k8-C8-V5	239	239	239	239
P-n20-k2-C10-V2	154	154	154	154
P-n21-k2-C11-V2	160	160	160	160
P-n22-k2-C11-V2	162	162	162	162
P-n55-k15-C28-V8	545.3	555	565	558

Table 2. Computational results on small and medium instances with $\theta = 3$

Instance	*LB* [2]	*ALNS* [2]	*ITS* [5]	*MA*
A-n32-k5-C11-V2	386	386	386	386
A-n33-k5-C11-V2	315	318	315	315
A-n34-k5-C12-V2	419	419	419	419
A-n38-k5-C13-V2	367	367	367	367
A-n63-k9-C21-V3	625.6	642	643	642
B-n31-k5-C11-V2	356	356	356	356
B-n34-k5-C12-V2	369	369	369	369
B-n35-k5-C12-V2	501	501	501	501
B-n45-k5-C15-V2	410	422	410	410
B-n50-k7-C17-V3	393	393	393	393
P-n16-k8-C6-V4	170	170	170	170
P-n19-k2-C7-V1	111	111	111	111
P-n20-k2-C7-V1	117	117	117	117
P-n40-k5-C14-V2	213	213	213	213
P-n50-k10-C17-V4	292	292	292	292

From the summary Tables 1 and 2, we can observe that our memetic algorithm performs slightly better in comparison to all other approaches for solving the considered both small and medium size instances. In one instance *P-n55-k15-C28-V8* the solution returned by our algorithm exhibits a gap of 0.01 % with the best known solution provided by the ALNS and for the rest of the instances for which the optimal solution is known we solved them optimally with our algorithm.

Regarding the computational times, it is difficult to make a fair comparison between algorithms, because they have been evaluated on different computers and they are implemented in different languages. The running time of our MA is proportional with the number of generations. From the computational exper-

iments, it seems that 10^4 generations are enough to explore the solution space of the GVRP. Our proposed heuristic algorithm seems to be slower than ALNS and comparable with ITS. Therefore we can conclude that our approach will be appropriate when the execution speed is not critical.

5 Conclusions

In this paper, we developed a memetic algorithm for solving the GVRP. The proposed heuristic integrates a number of original features: we combine a genetic programming with a powerful local search procedure, our local search procedure consists of five local search heuristics, their diversity and power being an important factor for solving successfully the GVRP.

The preliminary experimental results show that our algorithm is robust and compares favorably with all known heuristic approaches to the problem in terms of solution quality. In the future, we plan to asses the the generality and scalability of the proposed hybrid heuristic by testing it on more instances.

Acknowledgments. This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS - UEFISCDI, project number PN-II-RU-TE-2011-3-0113.

References

1. Baldacci, R., Bartolini, E., Laporte, G.: Some applications of the generalized vehicle routing problem. Journal of the Operational Research Society, 61 (7), 1072-1077 (2010)
2. Bektas, T., Erdogan, G., Ropke, S.: Formulations and Branch-and-Cut Algorithms for the Generalized Vehicle Routing Problem. Transportation Science, 45(3), 299-316 (2011)
3. Ghiani, G., Improta, G.: An efficient transformation of the generalized vehicle routing problem. European Journal of Operational Research, 122, 11-17 (2000)
4. Fischetti, M., Salazar, J.J, Toth, P.: A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Operations Research, 45, 378-394 (1997)
5. Moccia, L., Cordeau, J-F., Laporte, G.: An incremental neighborhood tabu search heuristic for the generalized vehicle routing problem with time windows. Journal of the Operational Research Society, 63(2), 232-244 (2012)
6. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Caltech Concurrent Computation Program, Report 826 (1989)
7. Pop, P.C., Kara, I., Horvat Marc, A.: New Mathematical Models of the Generalized Vehicle Routing Problem and Extensions, Applied Mathematical Modelling, 36(1), 97-107 (2012)
8. Pop, P.C., Matei, O., Pop Sitar, C., Chira, C.: A genetic algorithm for solving the generalized vehicle routing problem. Lecture Notes in Artificial Intelligence, 6077, 119-126 (2010)
9. Pop, P.C., Pintea, C., Zelina, I., Dumitrescu, D.: Solving the Generalized Vehicle Routing Problem with an ACS-based Algorithm. American Institute of Physics, 1117, 157-162 (2009)

A Modal Type System for Error Handling

Giuseppe Primiero

Department of Computer Science
Middlesex University
United Kingdom

Abstract. A modal type system for distributed computing is introduced which defines incorrect output states by mistakes and failures, followed by appropriate resolution strategies. Structural properties for error expressions and error reduction are shown.

1 Introduction

Programs as instances of algorithmic processes are mathematical structures embedded and executed in the real world and as such are not void of errors. Error handling is a major issue for both formal methods and applications with important consequences in the design of distributed systems, security, fault prediction, formal verification, repair strategies. From an engineering viewpoint, fault tolerance and handling are well-studied notions since the 70's,[1]. The formal study of computational errors in logical systems is, on the other hand, far less extensive. Propositional Dynamic Error Logic[2] offers a semantic interpretation for formulas and programs in a labelled transition system including an error state. Given its classical dynamic setting, a system incurring in a error state is not provided with any means for recovery. In a type system, the standard rule for falsehood expresses instructions to abort inconsistent processes, again with no recovery procedure. Our interest lies in three tasks for a type system:

1. to define minimal (in-)correctness conditions for reachable states;
2. to analyse errors in view of the different conditions breaches;
3. to design formal strategies to identify and correct errors.

We build on the recent extensive work on constructive modalities ([7, 8]) with applications to type theories ([9, 10]) and distributed systems (see e.g. [11–15]) made possible by the interpretation of proofs-as-programs under the Curry-Howard isomorphism. We build on previous work for a multi-modal polymorphic system ([16], [18]) where truth and validity of terms are used to interpret call-by-value and call-by-name operations in distributed computing. A calculus to reason about error states and their resolution is given in three steps. First, we formulate a language with locally indexed processes by functions for locally valid code

[1] See [1], [2], [3], [4], [5].
[2] PDEL is an extension of Propositional Dynamic Logic PDL, see [6].

and global values: errors can arise in executing them at locations where they are not valid. Second, we enrich the language with the means for mobility of terms and code via modal-style functions, for global and local validation from and to addresses: then errors can arise in fetching and sending data at wrong locations. Finally, we internalize these two main kinds of errors: functions for mistakes as conceptual, semantic or syntactic errors; and functions for failures as procedural, mobility-related errors.[3]

2 $\mathtt{ML}^{\Box\Diamond \mathit{err}}$: A Type System with Error States

$\mathtt{ML}^{\Box\Diamond\mathtt{err}}$ is a modal type system for mobile distributed programming with error states and handling procedures, inspired by an extension of a standard Martin-Löf type theory with judgemental modalities, avoiding dependent types.

Definition 1 (Syntax). *The syntax of* $\mathtt{ML}^{\Box\Diamond\mathtt{err}}$ *is defined by the following alphabet:*

$\mathtt{Programs}\ \Pi := x_i \mid a_i$, *for* $i \in \mathtt{I}$
$\mathtt{Specifications}\ \Sigma := \alpha \mid \bot \mid \alpha \times \beta \mid \alpha + \beta \mid \alpha \to \beta \mid \alpha \supset \beta$
$\mathtt{Locations}\ \mathtt{I} := 1 < \cdots < n$
$\mathtt{Operations}\ \Phi := exec(\alpha) \mid run_i(\alpha) \mid run_{i\cup j}(\alpha \cdot \beta) \mid run_{i\cap j}(\alpha \cdot \beta)$
$\mid synchro_j(\beta(exec(\alpha)))$, *where* $\cdot \in \{+, \times\}$
$\mathtt{Error\ Functions}\ \mathtt{H} := fail@_i(\tau) \mid mistake(\tau)$
$\mathtt{Concurrency\ Functions}\ \mathtt{C} := access@_i(t{:}\tau \setminus \bot) \mid \phi(\tau)WITH(\Pi \cup I \cup \Gamma), \phi \in \Phi$
$\mathtt{Data\ Stacks}\ (\mathtt{Contexts}) := \Gamma_i, \Delta_i \mid \circ_i \Gamma, \circ \in \{\Box, \Diamond\}$
$\mathtt{Remote\ Operations} := GLOB(\Box_{i\cup j}\Gamma, \alpha) \mid BROAD(\Diamond_{i\cap j}\Gamma, \alpha)$
$\mathtt{Portable\ Code} := RET(\Gamma_{i\cup j}, \alpha) \mid SEND(\Gamma_{i\cap j}, \alpha)$

Program terms (Π) include variables (x_i) to refer to local code and constants (a_i) to refer to globally safe values. Indices ($\mathtt{I}$) collect strictly ordered locations for execution of programs in a network. Types of programs (Σ) are output values defined recursively from the atomic type α using 'connectives' as functions. Local codes and global values define the two main operations in Φ by rule constructions: $exec(\alpha)$ is the output value generated by constant construction, is globally valid and can be called by name; $run_i(\alpha)$ is the local output value at i generated by variable construction, is locally bounded and can only be called by value. Compound code execution at ordered intersection of locations $i \cap j$ is given by $\times$; at union of locations $i \cup j$ by $+$; code functional composition ($\supset$) composes the outputs of two derivable *run* to their intersection at locations; values composition ($\to$) transforms a local *run* and a global *exec* into each other; *synchro* associates a value of a *run* with an *exec* function. Contexts express data stacks for program execution with valid code (true assumptions) $\Gamma_i := \{run_i(\alpha), \ldots, run_i(\nu) \mid x_1 : \alpha, \ldots, y_i : \nu\}$ and safe values (valid assumptions) $\Delta_i := \{exec(\alpha), \ldots, exec(\nu) \mid a_1 : \alpha, \ldots, n_i : \nu\}$. Modal contexts express

[3] This formal treatment is based on the taxonomy of errors for information systems with procedural semantics introduced in [17].

the occurrence of run and $exec$ functions: $\Box_i\Gamma = \{exec(\alpha) \mid \text{ for all } a_i : \alpha \in \Gamma\}$ is obtained from a stack of safe values only, and $\Box_i\Gamma \vdash \alpha$ declares code correctness of value α with complete instructional informations at run-time in Γ, all available from i but valid at *any* other location accessible from i; $\Diamond_i\Gamma = \{\circ_i(\alpha) \mid \circ \in \{\mathtt{\Phi}\}\}$ is inferred from a stack of valid code so that $\vdash run_i(\alpha)$ for at least one $x_i : \alpha \in \Gamma$, and $\Diamond_i\Gamma \vdash \alpha$ declares code correctness of value α with complete instructional informations at run-time in Γ, accessible only locally at i. Mobility rules are added in the form of introduction and elimination rules for transmission operators that allow to express at which reachable locations in a network a program executes and terminates. $GLOB$ works as a remote procedure call for safe values with RET as the corresponding declaration of portable value; $BROAD$ is the remote procedure call for valid code, with $SEND$ the corresponding portable rule. Concurrency functions $\mathtt{C}$ include: *access* as a command for program ($t \in \Pi$) accessibility at a location; and function execution ($\phi \in \mathtt{\Phi}$) on values ($\tau \in \Sigma$) with ($WITH$) an indexed, eventually compound program. Error functions $\mathtt{H}$ include *failures* for syntax errors on local processes and *mistakes* for semantic errors global on a specification.

Definition 2 (Typing Rules). *The set of typing rules for execution and mobility of code and values is:*

$$\frac{}{\Delta_i, a_i : \alpha \vdash exec(\alpha)}\ Global$$

$$\frac{\Delta_i, a_i : \alpha \vdash exec(\alpha)}{\Delta_i \vdash access@_i(a : \alpha)}\ @I \qquad \frac{\Delta_i; \Gamma_i \vdash access@_i(a : \alpha)}{\Delta_i; \Gamma_i, x_i : \alpha \vdash run_i(\alpha)}\ @E$$

$$\frac{\Delta_i; \Gamma_i \vdash \phi(\alpha)}{\Delta_i \vdash \phi(\alpha) WITH(t, \Gamma)}\ WITH\text{-}I \quad \frac{\Delta_i \vdash \phi(\alpha) WITH(t, \Gamma) \qquad \Delta_i; \Gamma \vdash t : \alpha}{\Delta_i; \Gamma, t : \alpha \vdash \phi(\alpha)}\ WITH\text{-}E$$

$$\frac{a_i : \alpha \vdash exec(\alpha) \qquad b_j : \beta \vdash exec(\beta)}{\vdash run_{i \cap j}(\alpha \times \beta)}\ I\times$$

$$\frac{\vdash run_{i \cap j}(\alpha \times \beta)}{\vdash exec(\alpha)}\ E1\times \qquad \frac{\vdash run_{i \cap j}(\alpha \times \beta)}{\vdash exec(\beta)}\ E2\times$$

$$\frac{a_i : \alpha \vdash exec(\alpha)}{\vdash run_i(\alpha + \beta)}\ I + (l) \qquad \frac{b_j : \beta \vdash exec(\beta)}{run_j(\alpha + \beta)}\ I + (r)$$

$$\frac{\vdash run_{i \cup j}(\alpha + \beta) \qquad run_i(\alpha) \vdash c_k : \gamma \qquad run_j(\beta) \vdash c_k : \gamma}{\vdash run_{i \cap j \cap k}(\gamma)}\ E+$$

$$\frac{\vdash run_i(\alpha) \qquad x_i : \alpha \vdash run_j(\beta)}{run_{i \cap j}(\alpha \supset \beta)}\ I \supset \quad \frac{\vdash run_{i \cap j}(\alpha \supset \beta) \qquad x_i : \alpha}{x_i : \alpha \vdash run_j(\beta)}\ E \supset$$

$$\frac{\vdash exec(\alpha) \qquad a_i : \alpha \vdash exec(\beta)}{\vdash run_{i \cup j}(\alpha \to \beta)}\ I \to \quad \frac{\vdash run_{i \cup j}(\alpha \to \beta) \qquad a_i : \alpha}{exec(\alpha) \vdash exec(\beta)}\ E \to$$

$$\frac{\vdash run_{i \cap j}(\alpha \supset \beta) \qquad a_i : \alpha}{\vdash synchro_j(\beta(exec(\alpha)))}\ Synchro$$

$$\frac{\Gamma_i, x_j : \alpha \vdash run_j(\alpha) \qquad \Box_i \Gamma, x_j(a_j) : \alpha \vdash exec(\alpha)}{GLOB(\Box_{i \cup j} \Gamma, \alpha)}\ RPC1$$

$$\frac{\Gamma_i, x_j : \alpha \vdash run_j(\alpha) \qquad \Diamond_i \Gamma \vdash run_j(\alpha)}{BROAD(\Diamond_{i \cap j} \Gamma, \alpha)}\ RPC2$$

$$\frac{\Box_i \Gamma, a_j : \alpha \vdash exec(\alpha) \qquad GLOB(\Box_{i \cup j} \Gamma, \alpha)}{RET(\Gamma_{i \cup j}, \alpha)}\ PORT1$$

$$\frac{\Box_i \Gamma, x_j : \alpha \vdash run_{i \cap j}(\alpha) \qquad BROAD(\Diamond_{i \cap j} \Gamma, \alpha)}{SEND(\Gamma_{i \cap j}, \alpha)}\ PORT2$$

Global generates an output value under valid assumptions or safe values. The @-operator tells how to access and run locally code for which safe value is available. The other concurrency function $WITH$ simply associate the *run* or *exec* function on value with the required program term and stack. Connectives $+, \times$ work as expected with values and codes. An instance of $\mathrm{I}\supset$ with $run_i(\alpha)$ and $x_i(\alpha) \vdash run_j(\alpha)$, allows to move from run_i to $run_{i \cap j}$. To move from run_i to $run_{i \cup j}$, one needs the intermediate step of obtaining an $exec(\alpha)$ from $run_i(\alpha)$, thus inducing $run_{i \cup j}(\alpha \to \alpha)$. $RPC1$ says that if all valid code for the execution of a program of type α is available, then α is globally valid with the relevant stack. The corresponding elimination $PORT1$ starts from a similarly derived value to decompose its valid locations. $RPC2$ says that if execution at i of a program for α requires code bounded to address j, then resources at the intersection of i, j are needed for any execution. The corresponding elimination $PORT2$ starts from a similarly derived valid code to infer its variable constructor for local validity of α. Error states are obtained either by a syntax error at execution of a valid program; or by a semantic error related to the (possibly correct) execution of a program. Both induce an invalid $\langle t_i, \tau \rangle \mid t \in \Pi; i \in \mathtt{I}; \tau \in \Sigma$ pair. Error functions have the following intended meaning:

- $fail@_i(\tau)$ is the state of the system where a program $t : \tau \setminus \bot$ induces a failure when accessed at index i;
- $mistake(\tau)$ is the state of the system where reference to specification τ induces an error.

Because ϕ does not range over $\mathtt{H}$ in $WITH$, the mistake function will not iterate. Similarly, in $access@$, the term t cannot be typed in $\bot$. This is required to obtain error reducibility formulated by Theorem 4.

2.1 Mistakes

Mistakes are semantic errors caused by missing resources for a given pair $\langle t_i, \tau \rangle$. In the following, we consider two formulations, with respectively $\phi := (exec)$ and $\phi := (run_i)$. In the first case, the system refers to a $\tau \in \Sigma$ for which no running $t \in \Pi$ has been defined. This is for example the case of a program that includes in its specification design a sub-process for which no routine has been defined. Consequently, any further specification υ depending on τ will induce an error state.

$$\frac{\Delta_i; \Gamma_i \vdash run_i(\tau \to \bot) \qquad \Delta_i; x_i : \tau \vdash run_i(\upsilon)}{\Gamma_i; \Delta_i \vdash mistake(exec(\upsilon)WITH(\tau))} \text{ Mistake1}$$

In the second case, given a $\tau \in \Sigma$, the corresponding $t \in \Pi$ is not reducible to any other valid one in the system. This is the case where a routine for the called specification exists, but is ill-defined. Correspondingly, matching any υ with any running t is ill-defined.

$$\frac{\Delta_i; \Gamma_i \vdash run_i(t \supset \bot) \qquad \Gamma_i, x_i : \tau \vdash run_i(\upsilon)}{\Gamma_i; \Delta_i \vdash mistake(run_i(\upsilon)WITH(t_i))} \text{ Mistake2}$$

2.2 Failures

Failures are syntax errors related to the use, accessibility and retrieval of physical resources. To reason about them we use the $access@_i$ function and generate failure of mobility rules. The first case of failure is given by a program accessing wrong resources (possibly at the right location). An example would be the choice of inappropriate procedural steps, like rules in a logical derivation. In this case we consider mobility rules for overall safe code that do not call upon the required resources. The operation involved is $PORT1$:

$$\frac{\Box_i \Gamma, t_j : \tau \vdash exec(\upsilon) \qquad GLOB(\Box_{i \cup j} \Gamma, \tau)}{\Box_i \Gamma, access@_j(t' : \tau') \vdash fail@_{i \cup j}(\upsilon) \quad (t' \neq t; \tau' \neq \tau)} \text{ FailPort1}$$

The first premise declares the required resources, the second the allowed mobility, while the conclusion states failure in view of renamed resources. Computationally, this rule is very expensive: to be complete it requires declaration for every $t' : \tau'$ that does not reduce to $t_j : \tau$.

The other case of failure is caused by programs accessing (possibly correct) data at wrong locations. It satisfies cases of failing calls on local resources. We refer here to program execution where code mobility is not overall safe in remote procedure calls and code portability. The rule involved is $RPC2$:

$$\frac{\Box_i \Gamma; x_j : \tau \vdash run_{i \cap j}(\upsilon) \qquad BROAD(\Diamond_{i \cap j} \Gamma, \tau)}{\Diamond_i \Gamma, access@_{k > j}(t : \tau) \vdash fail@_{i \cap j}(\upsilon)} \text{ FailPort2}$$

The first premise declares the required local resources, the second the allowed mobility, while the conclusion states failure in view of non-accessible locations.

3 Error Handling Mechanisms

For each error and failure function rule, we define an appropriate handling mechanism.

$$\frac{\Gamma_i \vdash mistake(exec(\upsilon)WITH(\tau)) \qquad \Gamma_i, t_j : \tau' \vdash exec(\tau')}{\Gamma_i \vdash synchro_j(\upsilon(exec([\tau/\tau'])))} \text{ HandleMistake1}$$

$$\frac{\Gamma_i \vdash mistake(run_i(\upsilon)WITH(t_i)) \qquad \Gamma_i, [t_i/x_j] : \tau \vdash run_i(\upsilon)}{\Gamma_i \vdash run_{j \cap i}(\tau \supset \upsilon)} \text{ HandleMistake2}$$

$$\frac{\Box_i \Gamma, access@_j(t' : \tau') \vdash fail@_{i\cup j}(\upsilon) \qquad RET(\Gamma_{i\cup j}, \tau) \qquad \tau \neq \tau'}{\Box_i \Gamma, [t'_j : \tau' / t_j : \tau] \vdash exec(\upsilon)} \text{ HandleFailPort1}$$

$$\frac{\Diamond_i \Gamma, access@_{k>j}(t : \tau) \vdash fail@_{i\cap j}(\upsilon) \qquad SEND(\Gamma_{i\cap j}, \tau)}{\Diamond_i \Gamma, [t_k / x_j] : \tau \vdash run_{i\cap j}(\upsilon)} \text{ HandleFailPort2}$$

For wrong typing, we require binding under a new typing element in the stack. For a well-defined specification using a wrong process, a new running one must be selected. For failures related to accessing wrong resources, we force access readdress. For failures related to accessing resources at wrong locations, we force access restriction. Substitutions are justified below by Theorem 2.

A final step induces the abort function from everywhere failing error handling mechanisms:

$$\frac{\Gamma_i \vdash mistake(exec(\upsilon) WITH(\tau)) \qquad GLOB(\Box_i \Gamma, mistake(exec(\tau(\upsilon))))}{\Gamma_i, \tau \vdash abort(\upsilon)} \text{ abortM1}$$

$$\frac{\Gamma_i \vdash mistake(run_i(\upsilon) WITH(t_i)) \qquad BROAD(\Diamond_i \Gamma, mistake(run_i(t_i(\upsilon))))}{\Gamma_i, t_i : \tau \vdash abort(\upsilon)} \text{ abortM2}$$

$$\frac{\Box_i \Gamma, access@_j(t' : \tau') \vdash fail@_{i\cup j}(\upsilon) \qquad \Gamma_{i\cup j} \vdash synchro(\upsilon(exec(\tau')))}{\Box_i \Gamma, t'_j : \tau' \vdash abort(\upsilon)} \text{ abortFP1}$$

$$\frac{\Diamond_i \Gamma, access@_{k>j}(t : \tau) \vdash fail@_{i\cap j}(\upsilon) \qquad \Gamma_{i\cap k} \vdash synchro(\upsilon(exec(\tau)))}{\Box_i \Gamma, t_k : \tau \vdash abort(\upsilon)} \text{ abortFP2}$$

4 Properties of Error Expressions

Theorem 1 (Structural Rules). *Structural Rules hold for any arbitrary derivation of* $\mathtt{ML}^{\Box\Diamond \mathtt{err}}$*, where in the following* $\phi, \psi, \rho \in \{x_i, a_i, access@_i\}$ *and* $\varphi \in \{run_i, exec, mistake, fail@_i, abort\}$*, as appropriate:*

1. *If* $\Delta_i; \Gamma_i, \phi(\alpha), \psi(\alpha) \vdash \varphi(\gamma)$ *then* $\Delta_i; \Gamma_i, \psi(\alpha), \phi(\alpha) \vdash \varphi(\gamma)$*;*
2. *If* $\Delta_i; \Gamma_i, \phi(\alpha) \vdash \varphi(\gamma)$ *then* $\Delta_i; \Gamma_i, \phi(\alpha), \phi(\alpha) \vdash \varphi(\gamma)$*;*
3. *If* $\Delta_i; \Gamma_i, \phi(\alpha), \psi(\alpha) \vdash \varphi(\gamma)$ *then* $\Delta_i; \Gamma_i, \rho(\alpha) \vdash \varphi(\gamma)[\phi/\rho; \psi/\rho]$*.*

Proof. By induction on the structure of the given derivations.

Theorem 2 (Substitution on terms). *The following substitutions hold:*

1. *If* $\Gamma_i, x_i : \tau; \Delta_i \vdash run_i(\upsilon)$*,* $\Gamma_i; \Delta_i \vdash mistake(exec(\upsilon) WITH(\tau))$ *and* $\Gamma_i \vdash synchro_j(\upsilon(exec(\tau')))$*, then* $\Gamma_i; \Delta, [x_i/t_j] : \tau' \vdash exec(\upsilon)$*.*
2. *If* $\Gamma_i, x_i : \tau; \Delta_i \vdash run_i(\upsilon)$*,* $\Gamma_i; \Delta_i \vdash mistake(exec(\upsilon) WITH(t)$ *and* $\Gamma_i \vdash run_{j\cap i}(\tau \supset \upsilon)$*, then* $\Gamma_i; \Delta, [x_i/t_j] : \tau \vdash run_{j\cap i}(\upsilon)$*.*
3. *If* $\Box_i \Gamma, t_j : \tau \vdash exec(\upsilon)$*,* $\Box_i \Gamma, acess@_j(t' : \tau') \vdash fail@_{i\cup j}(\upsilon)$ *and* $RET(\Gamma_{i\cup j}, \tau)$*, then* $\Box_i \Gamma, [t_i/t_j] : \tau \vdash exec(\upsilon)$*.*
4. *If* $\Box_i \Gamma, x_j : \tau \vdash run_{i\cap j}(\upsilon)$*,* $\Diamond_i \Gamma, access@_{k>j}(t : \tau) \vdash fail@_{i\cap j}(\upsilon)$ *and* $SEND(\Gamma_{i\cap j}, \tau)$*, then* $\Box_i \Gamma, [x_k/x_j] : \tau \vdash run_{i\cap j}(\upsilon)$*.*

Proof. By induction on the relevant derivation, using the structure of the mistake or failure rule.

Error functions and their handling rules can be seen as respectively introduction and elimination rules for $fail$ and $mistake$. To show that these are meaningful in a logical sense, detours obtained by the error introduction and its handling should generate a valid expression which would hold without errors. This proves local soundness.

$$\dfrac{\dfrac{\dfrac{\dfrac{\Delta_i; x_i : \tau \vdash run_i(\tau) \qquad exec(\tau) \vdash \bot}{\Delta_i \vdash run_i(\tau \to \bot)} \to\text{-I} \qquad \Delta_i, x_i : \tau \vdash run_i(v)}{\Delta_i; \Gamma_i \vdash mistake(exec(v) WITH(\tau)} \text{M1} \qquad \Gamma_i, t_j : \tau' \vdash exec(\tau')}{\Gamma_i; \Delta_i \vdash synchro_j(v(exec(\tau')))} \text{HM1}$$

$$\rightsquigarrow$$

$$\dfrac{\Gamma_i, x_j : \tau' \vdash run_{i \cap j}(\tau' \supset v) \qquad \Delta_i \vdash t_j : \tau'}{\Gamma_i; \Delta_i \vdash synchro_j(v(exec(\tau')))} \text{Synchro}$$

$$\dfrac{\dfrac{\dfrac{\Gamma_i; x_i : \tau \vdash run_i(\tau) \qquad run_i(\tau) \vdash \bot}{\Gamma_i \vdash run_i(\tau \supset \bot)} \supset\text{-I} \qquad \Gamma_i, x_i : \tau \vdash run_i(v)}{\Gamma_i \vdash mistake(run_i(v) WITH(t_i)} \text{M2} \qquad \Gamma_i, x_j : \tau \vdash run_i(v)}{\Gamma_i \vdash run_{j \cap i}(\tau \supset v)} \text{H2}$$

$$\rightsquigarrow$$

$$\dfrac{\Gamma_i \vdash run_j(\tau) \qquad \Gamma_i, x_j : \tau \vdash run_i(v)}{\Gamma_i \vdash run_{j \cap i}(\tau \supset v)} \text{I}\supset$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_i \vdash access@_i(t : \tau)}{\Gamma_i, x_i : \tau \vdash run_i(\tau)} @\text{E} \qquad \Box_i \Gamma, x_i / t_i : \tau \vdash exec(\tau)}{GLOB(\Box_{i \cup j} \Gamma, \tau)} \text{RPC1}}{\Box_i \Gamma, access@_i(t' : \tau') \vdash fail@_{i \cup j}(\tau)} \text{FP1} \qquad RET(\Gamma_{i \cup j}, \tau)}{\Box_i \Gamma, t_i : \tau \vdash exec(\tau)} \text{HFP1}$$

$$\rightsquigarrow$$

$$\dfrac{\Gamma_i \vdash t_i : \tau}{\Box_i \Gamma, t_i : \tau \vdash exec(\tau)} \text{Global}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Delta_i, t_i : \tau \vdash exec(\tau)}{\Delta_i \vdash access@_i(\tau)} @\text{I} \qquad \Delta_i, x_i : \tau \vdash run_i(\tau)}{BROAD(\Diamond_{i \cap j} \Gamma, \tau)} \text{RPC2}}{\Diamond_i \Gamma, access@_{j > i}(t : \tau) \vdash fail@_{i \cap j}(\tau)} \text{FP2} \qquad SEND(\Gamma_{i \cup j}, \tau)}{\Diamond_i \Gamma, x_j : \tau \vdash run_{i \cap j}(\tau)} \text{HFP2}$$

$$\rightsquigarrow$$

$$\dfrac{\Gamma_i \vdash run_i(\tau) \qquad x_i : \alpha \vdash run_j(\tau)}{\Diamond_i \Gamma, x_i : \tau \vdash run_{i \cap j}(\tau)} \text{I}\supset$$

Definition 3 ($\beta^{\mathtt{err}}$)**.** *The following resolve reductions hold in view of local soundness shown above:*

1. $\Delta_i; \Gamma_i \vdash mistake(exec(v)WITH(\tau))$ *and* $\Gamma_i, t_j : \tau' \vdash exec(\tau') \xrightarrow{\beta^{\mathtt{err}}} \Delta_i; \Gamma_i \vdash synchro(v(exec([\tau/\tau'])))$;
2. $\Gamma_i \vdash mistake(run_i(v)WITH(t_i)$ *and* $\Gamma_i, x_j : \tau \vdash run_i(v) \xrightarrow{\beta^{\mathtt{err}}} \Gamma_i \vdash run_{j\cap i}([t_i/t_j] : \tau \supset v)$;
3. $\Box_i\Gamma, access@_i(t' : \tau') \vdash fail@_{i\cup j}(\tau)$ *and* $RET(\Gamma_{i\cup j}, \tau) \xrightarrow{\beta^{\mathtt{err}}} \Box_i\Gamma, [t'_i/t_i] \vdash exec(\tau)$;
4. $\Diamond_i\Gamma, access@_{j>i}(t : \tau) \vdash fail@_{i\cap j}(t : \tau)$ *and* $SEND(\Gamma_{i\cup j}, \tau) \xrightarrow{\beta^{\mathtt{err}}} \Diamond_i\Gamma, [t_i/x_i] : \tau \vdash run_{i\cap j}(\tau)$.

The appropriate counterparts are expansions obtained by eliminations followed the relevant introductions, to show that handle-rules are not too weak. This proves local completeness.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma_i, t_j : \tau' \vdash exec(\tau')}\,GLOB \qquad \Gamma_i \vdash mistake(exec(v)WITH(\tau)}{\Gamma_i \vdash synchro_j(v(exec(\tau')))}\,\mathrm{HM1}}{\Gamma_i \vdash access@_j(\tau')}\,@I}{\Delta_i; \Gamma_i, x_j : \tau' \vdash run_j(\tau')}\,@E \qquad \Delta_i; \Gamma_i \vdash run_i(\tau \to \bot)}{\Delta_i; \Gamma_i \vdash mistake(exec(v)WITH(\tau))}\,\mathrm{M1}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_i \vdash access@_i(t : \tau)}{\Gamma_i, x_j : \tau \vdash run_i(\tau')}\,@E \qquad \Gamma_i \vdash mistake(run_i(v)WITH(t_i)}{\Gamma_i \vdash run_{j\cap i}(\tau \supset v)}\,\mathrm{HM2}}{\Gamma_i, x_j : \tau \vdash run_i(v)}\,\supset E \qquad \Gamma_i \vdash run_i(t \to \bot)}{\Delta_i; \Gamma_i \vdash mistake(run_i(v)WITH(t_i))}\,\mathrm{M2}$$

$$\dfrac{\dfrac{\dfrac{}{RET(\Gamma_{i\cup j}, \tau)}\,PORT1 \qquad \Box_i\Gamma, access@_j(t' : \tau) \vdash fail@_{i\cup j}(v)}{\Box_i\Gamma, t_j : \tau \vdash exec(v)}\,\mathrm{HFP1} \qquad GLOB(\Box_{i\cup j}\Gamma, v)}{\Box_i\Gamma, access@_j(t' : \tau') \vdash fail@_{i\cup j}(v)}\,\mathrm{FP1}$$

$$\dfrac{\dfrac{\dfrac{}{SEND(\Gamma_{i\cap j}, \tau)}\,PORT2 \qquad \Diamond_i\Gamma, access@_{k>j}(t : \tau) \vdash fail@_{i\cap j}(v)}{\Box_i\Gamma, x_j : \tau \vdash run_{i\cap j}(u : v)}\,\mathrm{HFP2} \qquad BROAD(\Diamond_{i\cap j}\Gamma, v)}{\Diamond_i\Gamma, access@_{j>i}(t : \tau) \vdash fail@_{i\cap j}(v)}\,\mathrm{FP2}$$

Definition 4 ($\eta^{\mathtt{err}}$)**.** *The following error expansions hold in view of the local completeness shown above:*

1. $\Delta_i, run(\tau \to \bot) \vdash run_i(v) \xrightarrow{\eta^{\mathtt{err}}} \Delta_i; \Gamma_i \vdash mistake(exec(v)WITH(\tau)$;
2. $\Gamma_i, run_i(t \supset \bot) \vdash run_i(v) \xrightarrow{\eta^{\mathtt{err}}} \Delta_i; \Gamma_i \vdash mistake(run_i(v)WITH(t_i)$;

3. $\Box_i \Gamma, GLOB(\Box_{i\cup j}\Gamma, \tau) \vdash run_{i\cup j}(\upsilon) \xrightarrow{\eta^{\mathrm{err}}} \Box_i \Gamma, access@_j(t' : \tau') \vdash fail@_{i\cup j}(u : \upsilon)$;
4. $\Box_i \Gamma, BROAD(\Diamond_{i\cap j}\tau) \vdash run_{i\cap j}(\upsilon) \xrightarrow{\eta^{\mathrm{err}}} \Diamond_i \Gamma, access@_{k>j}(t : \tau) \vdash fail_{i\cap j}(u : \upsilon)$.

In the following, we use respectively ERR for an expression obtained by $M1, M2, FP1$ or $FP2$; and RES for one obtained by $HM1, HM2, HFP1$ or $HFP2$:

Theorem 3 (Error Reduction and Expansion). *Subject reduction and expansion for error expressions hold as follows:*

- *If* $\Delta_i; \Gamma_i \vdash ERR$ *and* $ERR \xrightarrow{\beta^{\mathrm{err}}} RES$, *then* $\Delta_i; \Gamma_i \vdash RES$.
- *If* $\Delta_i; \Gamma_i \vdash ERR$ *and* $ERR \xrightarrow{\eta^{\mathrm{err}}} ERR'$, *then* $\Delta_i; \Gamma_i \vdash ERR'$.

Proof. For $\xrightarrow{\beta^{\mathrm{err}}}$, use substitution properties on the relevant expression of RES. For $\xrightarrow{\eta^{\mathrm{err}}}$, reconstruct the derivation of ERR'.

β-reduction reduces ERR-expressions to RES-expressions, possibly after a finite number of $\xrightarrow{\eta^{\mathrm{err}}}$ operations, until it cannot be longer reduced to a sub-expression of error-type. This is called an irreducible resolution expression, which can still contain a traditional redex of a non-error type.

Theorem 4 (Irreducible RES-expression). *If* $\Delta_i; \Gamma_i \vdash ERR$ *and* $ERR \xrightarrow{\beta^{\mathrm{err}}} RES$ *and there is no* RES' *such that* $RES \xrightarrow{\beta^{\mathrm{err}}} RES'$, *then* RES *is an irreducible resolution expression.*

Proof. The proof proceeds by error reduction. By inversion, consider the last inference step (error rule) in the derivation. Because RES has no further error reduction available, it will be obtained by one of $HM1, HM2, HFP1, HFP2$ and because iteration of error functions is not allowed, this term will have no subterm which is introduced by $M1, M2, FP1, FP2$. Hence it is an irreducible RES-expression.

Acknowledgements

This research was conducted while the author was Post-Doctoral Fellow of the Research Foundation Flanders (FWO) at the Centre for Logic and Philosophy of Science, Ghent University Belgium. He gratefully acknowledges the financial support.

References

1. Goodenough, J.: Exception handling: issues and a proposed notation. Commun. ACM **8** (1975) 683–696
2. Cristian, F.: Exception handling and software fault-tolerance. IEEE Transactions on Computers **C-31** (1982) 531–540
3. Rennels, D.: Fault-tolerant computing - concepts and examples. IEEE Transactions on Computers **C-33** (1984) 1116–1129
4. Cristian, F.: A rigorous approach to fault-tolerant programming. IEEE Transactions on Software Engineering **SE-11** (1985) 23–31
5. Cristian, F.: Understanding fault-tolerant systems. Commun. ACM **34** (1991) 56–78
6. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge, MA (2000)
7. Bierman, G., de Paiva, V.: On an intuitionistic modal logic. Studia Logica (65) (2000) 383–416
8. Alechina, N., Mendler, M., de Paiva, V., Ritter, E.: Categorical and Kripke Semantics for Constructive S4 Modal Logic. In: Proceedings of the 15th International Workshop on Computer Science Logic. Volume 2142 of Lecture Notes In Computer Science. (2001) 292 – 307
9. Pfenning, F., Davies, R.: A judgemental reconstruction of modal logic. Mathematical Structures in Computer Science **11** (2001) 511–540
10. Nanevski, A., Pfenning, F., Pientka, B.: Contextual modal type theory. ACM Transactions on Computational Logic **9**(3) (2008) 1–48
11. Moody, J.: Modal logic as a basis for distributed computation. Technical Report CMU-CS-03-194, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, USA (2003)
12. Davies, R., Pfenning, F.: A modal analysis of staged computation. Journal of the ACM **48**(3) (2001) 555–604
13. Murphy, T.: Modal Types for Mobile Code. PhD thesis, School of Computer Science, Carnegie Mellon University (2008) CMU-CS-08-126.
14. Murphy, T., Crary, K., Harper, R. In: Type-Safe Distributed Programming with ML5. Volume 4912 of Lectures Notes in Computer Science. Springer Verlag (2008) 108–123
15. Jia, L., Walker, D.: Modal Proofs as Distributed Programs. In: Programming Languages and Systems, ESOP2004. Volume 2986 of Lectures Notes in Computer Science., Springer Verlag (2004)
16. Primiero, G.: A multi-modal type system and its procedural semantics for safe distributed programming. In: Intuitionistic Modal Logic and Applications Workshop (IMLA11), Nancy (2011)
17. Primiero, G.: A taxonomy of errors for information systems. Minds & Machines (2013)
18. Primiero, G.: A contextual type theory with judgemental modalities for reasoning from open assumptions. Logique & Analyse **220** (2012) 579–600

The double negation translation in Nonstandard and Constructive Analysis

Sam Sanders

[1] Department of Mathematics, Ghent University
[2] Munich Center for Mathematical Pilosophy, Ludwig-Maximilian-Universität
sasander@cage.ugent.be

Abstract. We present a natural interpretation $\mathbb{B}$ of Errett Bishop's *Constructive Analysis* ([2]) inside classical Nonstandard Analysis ([10–14]) in the spirit of the 'reuniting the antipodes' program ([15]). The two fundamental notions *proof* and *algorithm* of Constructive Analysis are interpreted by $\mathbb{B}$ as *Transfer* and *Ω-invariance*. Intuitively, an object is *Ω-invariant* if it is independent of the *choice* of infinitesimal used in its definition. As the latter is the way infinitesimals are used in e.g. physics and engineering, Ω-invariance constitutes an alternative 'natural' model of computation directly inspired by these disciplines. Furthermore, we discuss the connection between $\mathbb{B}$ and the well-known *double negation translation* ([5, Chapter V]). In particular, we show that this translation is compatible with $\mathbb{B}$, and the results in this paper provide additional evidence for the faithfulness or naturalness of $\mathbb{B}$. Finally, we suggest an alternative version of the double negation translation based on $\mathbb{B}$.

1 Introduction

1.1 Interpreting Constructive Analysis inside Nonstandard Analysis

In [10, 12], the author introduces an interpretation of Errett Bishop's *Constructive Analysis* (or BISH, [2, 3]) inside a system $\mathbb{NSA}$ of Nonstandard Analysis. This interpretation, called $\mathbb{B}$, is 'natural' and 'faithful' in the following sense:

(i) Non-constructive principles (LPO, LLPO, MP, etc) are interpreted by $\mathbb{B}$ as instances of the *Transfer Principle* from Nonstandard Analysis not available in $\mathbb{NSA}$. This even seems to hold for the semi-constructive WMP and BD-N.
(ii) Non-constructive principles still satisfy the well-known equivalences and implications of *Constructive Reverse Mathematics* ([8, 9]) under $\mathbb{B}$.
(iii) Due to the absence of Markov's principle, the primitive notion of *algorihm* on $\mathbb{N}$ is weaker than that of recursive function in BISH, i.e. not all Δ_1^0-formulas are decidable. This property is preserved by $\mathbb{B}$.

The central notions of Constructive Analysis, i.e. *proof* and *algorithm*, are interpreted by $\mathbb{B}$ as *Transfer* ($\mathbb{T}$) and *Ω-invariance*. The latter are natural and elegant notions inspired by Mathematics and Physics; We sketch them now.

The *Brouwer-Heyting-Kolmogorov* interpretation used in BISH ([4]) is limited to formulas that come with proofs. Similarly and intuitively, we only consider formulas A in $\mathbb{NSA}$ which satisfy $A \leftrightarrow {}^*A$, where '$*$' is the star morphism from Nonstandard Analysis. The formula '$A \leftrightarrow {}^*A$', abbreviated as '$A \in \mathbb{T}$', is the general form of the *Transfer Principle* of Nonstandard Analysis. As $\mathbb{NSA}$ is a weak system, the Transfer Principle is not available. We also read '$A \in \mathbb{T}$' as 'A satisfies Transfer'.

The notion of *Ω-invariance* is defined as follows. The set of *infinite* or *nonstandard* numbers Ω is ${}^*\mathbb{N} \setminus \mathbb{N}$, where $\mathbb{N}$ is the set of *standard* or *finite* numbers and ${}^*\mathbb{N}$ is its nonstandard extension.

Definition 1 (Ω-invariance) *Let $\psi(n, m)$ be Δ_0 and fix $\omega \in \Omega$. The formula $\psi(n, \omega)$ is* Ω-invariant *if*

$$(\forall n \in \mathbb{N})(\forall \omega, \omega' \in \Omega)(\psi(n, \omega) \leftrightarrow \psi(n, \omega')). \tag{1}$$

The generalization of Ω-invariance to formulas $\psi(\boldsymbol{x}, \omega)$ is immediate. Note that an Ω-invariant object does not depend on the *choice* of the infinite number used in its definition. This is exactly the way infinitesimals are used in practice (Physics, Engineering, Applied Mathematics).

Initial results towards $\mathbb{B}$ can be found in [11–14]. In Section 2, we introduce $\mathbb{B}$ and $\mathbb{NSA}$ in detail. Clearly, the above items (i)-(iii) suggest that $\mathbb{NSA}$ captures BISH quite well, albeit indirectly. In Section 3, we show that $\mathbb{B}$ is compatible with the Gödel-Gentzen *double negation translation*, introduced next. In particular, the results in this paper suggest a tight correspondence between BISH and $\mathbb{NSA}$. We also introduce $\mathcal{M}$, an alternative version of the double negation translation which exploits the nonstandard nature of $\mathbb{NSA}$. Finally, the study of the double negation translation leads to a number of small (but necessary) technical improvements to our interpretation $\mathbb{B}$, as discussed in Appendix B.

1.2 The Double Negation Translation

The Gödel-Gentzen *double negation translation* is the syntactical operation $\mathcal{N}$ defined in Definition 2 (See [5, Chapter V]). The translation $\mathcal{N}$ connects provability in intuitionistic and classic logic, as is clear from Theorem 3 below.

Definition 2 [Double negation translation]

1. If A is atomic, then $A^{\mathcal{N}}$ is $\neg\neg A$, and $(\neg A)^{\mathcal{N}}$ is $\neg(A^{\mathcal{N}})$.
2. $(A \wedge B)^{\mathcal{N}}$ is $A^{\mathcal{N}} \wedge B^{\mathcal{N}}$, and $(A \to B)^{\mathcal{N}}$ is $A^{\mathcal{N}} \to B^{\mathcal{N}}$.
3. $(A \vee B)^{\mathcal{N}}$ is $\neg\neg(A^{\mathcal{N}} \vee B^{\mathcal{N}}) \equiv \neg(\neg(A^{\mathcal{N}}) \wedge \neg(B^{\mathcal{N}}))$.
4. $\big[(\forall x)A(x)\big]^{\mathcal{N}}$ is $(\forall x)\big[A(x)\big]^{\mathcal{N}}$.
5. $\big[(\exists x)A(x)\big]^{\mathcal{N}}$ is $\neg\big[(\forall x)\neg[A(x)]^{\mathcal{N}}\big] \equiv \neg\neg\big[(\exists x)[A(x)]^{\mathcal{N}}\big]$.

Various versions of the following theorem can be found in [1, Chapter D.5] and [16, Section 3.8]. Here, 'intuitionistic logic' refers to the well-known *Brouwer-Heyting-Kolmogorov* (BHK) interpretation of the logical connectives ([4, 5]).

Theorem 3 *The formula A is provable in a system T using classical logic* if and only if *the formula $A^{\mathcal{N}}$ is provable in a system $T^{\mathcal{N}}$ using intuitionistic logic*

In light of (i)-(iii) in Section 1.1, the essential features of BISH are captured by $\mathbb{NSA}$ under the interpretation $\mathbb{B}$. Given this similarity, and as BISH is based on intuitionistic logic, we expect the following to hold:

$$\mathbb{NSA} + \mathbb{LPO} \vdash A \text{ if and only if } \mathbb{NSA} \vdash A^{\mathcal{N}}, \tag{2}$$

where $\mathbb{LPO}$ is $\mathbb{B}(\text{LPO})$. In particular, for a non-constructive principle X implied by LPO, we expect that $\mathbb{NSA}$ can prove $\mathbb{X}^{\mathcal{N}}$, where $\mathbb{X}$ is $\mathbb{B}(X)$. We investigate this claim in Section 3.

2 The interpretation $\mathbb{B}$ of BISH in $\mathbb{NSA}$

The system $\mathbb{NSA}$ is defined in Appendix A. Essentially, it is a nonstandard extension of RCA_0 with recursive comprehension replaced by comprehension for Ω-invariant formulas, called Ω-CA in Definitions 18 and 19. Also, there is an intermediate nonstandard extension ${}^{\diamond}\mathbb{N}$ between $\mathbb{N}$ and ${}^{*}\mathbb{N}$ providing the following additional structure: The numbers in ${}^{\diamond}\mathbb{N} \setminus \mathbb{N}$ are called 'small infinite' and the numbers in ${}^{*}\mathbb{N} \setminus {}^{\diamond}\mathbb{N}$ are called 'large infinite', while elements of ${}^{\diamond}\mathbb{N}$ are also called '$\diamond$-standard' or '$\diamond$-finite'. These 'levels of infinity' are essential for $\mathbb{B}$, as discussed in Remark 7.

The system $\mathbb{NSA}$ is rather weak, given its role as a base theory described in Section 1.1. Furthermore, Constructive Reverse Mathematics is mostly limited to objects of low type and this is reflected in the language of $\mathbb{NSA}$. Extending $\mathbb{NSA}$ with more induction or higher types is straightforward, as long as the Transfer Principle remains unprovable, as is clear from (i) and (ii) in Section 1.1.

The interpretation $\mathbb{B}$ from BISH to $\mathbb{NSA}$ takes a formula C in the language of BISH as input and produces a formula $\mathbb{B}(C)$ in the language of $\mathbb{NSA}$. The new symbols '$\mathbb{V}$', '$\Rrightarrow$', and '$\sim$' from Definition 5 are called 'hyperconnectives' and are the nonstandard counterparts of the constructive connectives.

Definition 4 [$\mathbb{B}$] Let A, B be formulas in the language of BISH.

1. For $A \in \Delta_0$, we have $\mathbb{B}(A) \equiv A$.
2. For formulas A, B, we have $\mathbb{B}(A \vee B) \equiv [\mathbb{B}(A) \mathbb{V} \mathbb{B}(B)]$.
3. For formulas A, B, we have $\mathbb{B}(A \wedge B) \equiv [\mathbb{B}(A) \wedge \mathbb{B}(B)]$.
4. For formulas A, B, we have $\mathbb{B}(A \to B) \equiv [\mathbb{B}(A) \Rrightarrow \mathbb{B}(B)]$.
5. For a formula A, we have $\mathbb{B}(\neg A) \equiv [\sim \mathbb{B}(A)]$.
6. For a formula A, we have[3] $\mathbb{B}\big((\exists x) A(x)\big) \equiv [(\exists x) \mathbb{B}(A(x))]$
7. For a formula A, we have $\mathbb{B}\big((\forall x) A(x)\big) \equiv [(\forall x) \mathbb{B}(A(x))]$

[3] By Definition 6, the BHK-existential quantifier is *not* interpreted as the classical existential quantifier, although one may get this impression from the definition of $\mathbb{B}$.

The following definition is part of classical Nonstandard Analysis. Hence, the connectives '$\to$', '$\wedge$', etc. have their usual *classical* meaning. The exact definition of '$A \in \mathbb{T}$' is given in Definition 6 below. Intuitively, '$A \in \mathbb{T}$' denotes an instance involving A of the Transfer Principle of Nonstandard Analysis.

Definition 5 [Hyperconnectives] Let A, B be formulas in the language[4] of $\mathbb{NSA}$.

1. The formula $A \Rrightarrow B$ is defined as $[A \wedge A \in \mathbb{T}] \to [B \wedge B \in \mathbb{T}]$.
2. The formula $A \between B$ is defined[5]

$$\psi(\boldsymbol{x}, \omega) \to [A(\boldsymbol{x}) \wedge A(\boldsymbol{x}) \in \mathbb{T}] \wedge \neg\psi(\boldsymbol{x}, \omega) \to [B(\boldsymbol{x}) \wedge B(\boldsymbol{x}) \in \mathbb{T}], \qquad (3)$$

   for formulas $A(\boldsymbol{x})$, $B(\boldsymbol{x})$ involving standard parameters $\boldsymbol{x}$.
3. The formula $\sim A$ is defined as $A \Rrightarrow 0 = 1$.

On a technical note, to make sure $A \Rrightarrow B$ is similar to the BHK-implication $A \to B$, the system $\mathbb{NSA}$ includes a standard part principle (or comprehension principle) for Ω-invariant formulas. Indeed, $A \wedge A \in \mathbb{T}$, i.e. the antecedent of $A \Rrightarrow B$, allows us to define new[6] Ω-invariant formulas involving A and Ω-CA converts these into standard objects used to show $B \wedge B \in \mathbb{T}$.

It is clear that our interpretation $\mathbb{B}$ from BISH to $\mathbb{NSA}$ hinges on the notation '$A \in \mathbb{T}$'. We now define the latter in $\mathbb{NSA}$. It should be noted that there is an asymmetry in $\mathbb{T}$ between the existential and universal quantifier, involving ${}^{\diamond}\mathbb{N}$. As explained in Remark 7, this is essential to obtain an interpretation of BISH.

The following formulas are called 'trivial for $\mathbb{T}$': Δ_0-formulas, $(\exists n \in \mathbb{N})\varphi(n)$, $(\exists n \in {}^{\diamond}\mathbb{N})\varphi(n)$, $(\forall n \in {}^{*}\mathbb{N})\varphi(n)$, $(\exists n \in \mathbb{N})(\forall m \in {}^{*}\mathbb{N})\varphi(n, m)$, and $(\exists n \in {}^{\diamond}\mathbb{N})(\forall m \in {}^{*}\mathbb{N})\varphi(n, m)$ for $\varphi \in \Delta_0$, and formulas unsuitable for Transfer, like $(\forall n \in \mathbb{N})(n \notin \Omega)$. Finally, define '$u$ is v-standard' as 'if v is standard (resp. $\diamond$-standard), then u is standard (resp. $\diamond$-standard).

Definition 6 [$\mathbb{T}$] For formulas A, we define $A \in \mathbb{T}$ as follows, where $\varphi \in \Delta_0$.

A	$A \in \mathbb{T}$
A is trivial for $\mathbb{T}$	$0 = 0$
$(\forall n \in \mathbb{N})\varphi(n)$	$(\forall n \in \mathbb{N})\varphi(n) \to (\forall n \in {}^{*}\mathbb{N})\varphi(n)$
$(\forall n \in {}^{\diamond}\mathbb{N})\varphi(n)$	$(\forall n \in {}^{\diamond}\mathbb{N})\varphi(n) \to (\forall n \in {}^{*}\mathbb{N})\varphi(n)$
$(\exists n \in {}^{*}\mathbb{N})\varphi(n)$	$(\exists n \in {}^{*}\mathbb{N})\varphi(n) \to (\exists n \in {}^{\diamond}\mathbb{N})\varphi(n)$
$(\exists n \in {}^{*}\mathbb{N})(\forall m \in {}^{*}\mathbb{N})\varphi(n, m)$	$(\exists n \in {}^{*}\mathbb{N})(\forall m \in {}^{*}\mathbb{N})\varphi(n, m) \to (\exists n \in {}^{\diamond}\mathbb{N})(\forall m \in {}^{*}\mathbb{N})\varphi(n, m)$
$(\exists n \in {}^{\diamond}\mathbb{N})(\forall m \in {}^{\diamond}\mathbb{N})\varphi(n, m)$	$(\exists n \in {}^{\diamond}\mathbb{N})(\forall m \in {}^{\diamond}\mathbb{N})\varphi(n, m) \to (\exists n \in {}^{\diamond}\mathbb{N})(\forall m \in {}^{*}\mathbb{N})\varphi(n, m)$
$(\exists n \in \mathbb{N})(\forall m \in \mathbb{N})\varphi(n, m)$	$(\exists n \in \mathbb{N})(\forall m \in \mathbb{N})\varphi(n, m) \to (\exists n \in \mathbb{N})(\forall m \in {}^{*}\mathbb{N})\varphi(n, m)$

[4] The language of $\mathbb{NSA}$ is introduced in Appendix A. It is a nonstandard version of $\mathcal{L}_2$, the language of second-order arithmetic.

[5] For general formulas involving $A(\boldsymbol{x}) \between B(\boldsymbol{x})$, see Definition 21.

[6] For instance, if $A \in \Pi^0_1$ is $(\forall n \in \mathbb{N})\varphi(n)$ and we have $A \wedge A \in \mathbb{T}$, then $(\forall n \leq \omega)\varphi(n)$ is Ω-invariant.

For $\varphi(n,z) \in \Delta_0$ and A as $(\forall n \in \mathbb{N})(\exists z \in Z)\varphi(n,z)$, the formula $A \in \mathbb{T}$ is

$$(\forall n \in \mathbb{N})(\exists z \in Z)\varphi(n,z) \to (\forall n \in {}^*\mathbb{N})(\exists z \in {}^*Z)[z \text{ is } n\text{-standard} \wedge \varphi(n,z)]. \quad (4)$$

For A given as $(\forall n \in \mathbb{N})(\exists m \in \mathbb{N})(\exists k \in \mathbb{N})\varphi(n,m) \in \Pi_3$, the formula $A \in \mathbb{T}$ is

$$A \to (\forall n \in {}^*\mathbb{N})(\exists m \in {}^*\mathbb{N})(\forall k \in {}^*\mathbb{N})[m \text{ is } n\text{-standard} \wedge \varphi(n,m,k)]. \quad (5)$$

A 'mixed' formula like $(\forall n \in \mathbb{N})(\forall m \in {}^\diamond\mathbb{N})\varphi(n,m)$ is treated as $(\forall n \in \mathbb{N})(\forall m \in \mathbb{N})\varphi(n,m)$. If A fits several of the above categories, then we chose the one with the lowest number of quantifier alternations to define $A \in \mathbb{T}$. If the number of alternations is the same but one category is 'trivial for Transfer', choose the other category. Otherwise, choose the existential version of A to define $A \in \mathbb{T}$.

Finally, $A \in \mathbb{T}$ where A is $(\exists \alpha \in 2^{\mathbb{N}})A(\alpha)$ is defined as follows.

$$A \in \mathbb{VT} \wedge (\exists \alpha \in 2^{\mathbb{N}})A(\alpha) \to (\exists \alpha \in 2^{\mathbb{N}})\big[A(\alpha) \wedge A(\alpha) \in \mathbb{T}\big], \quad (6)$$

where '$A \in \mathbb{VT}$' is called 'Disjunctive Transfer' and defined[7] as

$$(\forall \alpha, \beta \in 2^{\mathbb{N}})\big[A(\alpha) \vee A(\beta) \Rrightarrow A(\alpha) \mathbin{\mathbb{V}} A(\beta)\big]. \quad (7)$$

Disjunctive Transfer expresses that if one of two formulas $A(\alpha)$ and $A(\beta)$ satisfies Transfer, we can decide which one does. Thus, (6) tells us what the 'constructive' notion of existence means in $\mathbb{NSA}$: There is a classical witness and given two potential witnesses, we can pick out the right one, thanks to $\mathbb{VT}$. Note that this notion of existence is 'external' in the same way that Ω-invariance is an external, i.e. from the outside, notion of algorithm.

Remark 7 In intuitionistic logic, the following asymmetry between the universal and existential quantifier is present: We have $\neg[(\exists x)A(x)] \leftrightarrow (\forall x)\neg A(x)$, but only $\neg[(\forall x)A(x)] \leftarrow (\exists x)\neg A(x)$ in general. In other words, $\neg[(\forall x)A(x)]$ is weaker than $(\exists x)\neg A(x)$. Thanks to the introduction of ${}^\diamond\mathbb{N}$ and $\mathbb{T}$, we have for $A \in \Delta_0$ that $\sim[(\exists n \in \mathbb{N})A(n)] \Longleftrightarrow (\forall n \in \mathbb{N})\sim A(x)$, but $\sim(\forall n \in \mathbb{N})A(n) \Lleftarrow \Rrightarrow (\exists n \in {}^\diamond\mathbb{N})\sim A(n)$, which is *weaker* than $(\exists n \in \mathbb{N})\sim A(n)$, as $\mathbb{N} \subsetneq {}^\diamond\mathbb{N}$. Thus, the asymmetry in $\mathbb{T}$ between the existential and universal quantifier involving ${}^\diamond\mathbb{N}$ is *needed* to obtain intuitionistic logic, in particular the asymmetry between existential and universal quantification.

With the above definition of $\mathbb{T}$, it is possible to prove (i)-(iii) from Section 1.1. In particular, the numerous CRM-equivalences from Ishihara's overview paper ([8]) are proved with $\mathbb{NSA}$ as a base theory in [10, 12]. We now list two insightful examples of these results. For a non-constructive principle X rejected in BISH like LPO, LLPO, MP etc, we denote $\mathbb{B}(X)$ by $\mathbb{X}$.

Theorem 8 *In $\mathbb{NSA}$, the following are equivalent.*

1. Π_1-TRANS, *For all $\varphi \in \Delta_0$, $(\forall n \in \mathbb{N})\varphi(n) \to (\forall n \in {}^*\mathbb{N})\varphi(n)$.*

[7] For cosmetic reasons, we require that $\alpha \neq \beta$ in (7).

2. $\mathbb{LPO}$, *For all* $P \in \Sigma_1$, $P \vee \sim P$.
3. $\mathbb{LPR}$, $(\forall x \in \mathbb{R})(x > 0 \vee \sim(x > 0))$.
4. $\mathbb{MCT}$, *The monotone convergence theorem.*

In $\mathbb{NSA}$, *the following are equivalent.*

1. $\mathbb{LLPO}$, *For* $P, Q \in \Sigma_1$, $\sim(P \wedge Q) \Rightarrow \sim P \vee \sim Q$.
2. $\mathbb{LLPR}$, $(\forall x \in \mathbb{R})(x \geq 0 \vee x \leq 0)$.
3. $\mathbb{WKL}$, $(\forall n \in \mathbb{N})(\exists \alpha \in 2^{\mathbb{N}})(\overline{\alpha}n \in T) \Rightarrow (\exists \alpha \in 2^{\mathbb{N}})(\forall n \in \mathbb{N})(\overline{\alpha}n \in T)$.
4. $\mathbb{NIL}$, $(\forall x, y \in \mathbb{R})(xy = 0 \Rightarrow x = 0 \vee y = 0)$.

Note that a real number x is defined (in BISH and $\mathbb{NSA}$ alike) as a sequence of rationals $(q_n)_{n\in\mathbb{N}}$ satisfying $(\forall n, m \in \mathbb{N})(|q_n - q_{n+m}| < \frac{1}{2^n})$

3 The double negation translation in $\mathbb{NSA}$

3.1 Introduction

In this section, we provide the main results of this paper, namely that $\mathbb{NSA}$ proves $\mathbb{B}(X)^{\mathcal{N}}$ for non-constructive principles X like LPO, LLPO, MP, et cetera. By items (i) and (ii), $\mathbb{NSA}$ cannot prove $\mathbb{B}(X)$, as the latter imply Transfer Principles not available in $\mathbb{NSA}$. Thus, the results in this section suggest that $\mathbb{B}$ is compatible with $\mathcal{N}$, and furthermore provide additional evidence for the tight correspondence between BISH and $\mathbb{NSA}$ in light of Theorem 3. Finally, we also discuss the possibility of an alternative version of the double negation translation which exploits the nonstandard nature of $\mathbb{NSA}$. We will use the following (obvious in light of $\mathbb{B}$) version of the double negation translation in $\mathbb{NSA}$.

Definition 9

1. If A is atomic, then $A^{\mathcal{N}}$ is $\sim\sim A$.
2. $(A \wedge B)^{\mathcal{N}}$ is $A^{\mathcal{N}} \wedge B^{\mathcal{N}}$.
3. $(A \vee B)^{\mathcal{N}}$ is $\sim\sim(A^{\mathcal{N}} \vee B^{\mathcal{N}}) \iff \sim(\sim(A^{\mathcal{N}}) \wedge \sim(B^{\mathcal{N}}))$.
4. $(A \Rightarrow B)^{\mathcal{N}}$ is $A^{\mathcal{N}} \Rightarrow B^{\mathcal{N}}$.
5. $(\sim A)^{\mathcal{N}}$ is $\sim(A^{\mathcal{N}})$.
6. $\big[(\forall x)A(x)\big]^{\mathcal{N}}$ is $(\forall x)\big[A(x)\big]^{\mathcal{N}}$.
7. $\big[(\exists x)A(x)\big]^{\mathcal{N}}$ is $\sim\big[(\forall x)\sim[A(x)]^{\mathcal{N}}\big] \iff \sim\sim\big[(\exists x)[A(x)]^{\mathcal{N}}\big]$.

The hyperequivalences in items 3 and 7 of this definition are proved in Appendix B. We first discuss the definition of $\mathbb{T}$ in this context.

The idea behind the definition of $\mathbb{T}$ is clear from Definition 6: The formula $A \in \mathbb{T}$ is that instance of the Transfer Principle where the universal (resp. existential) quantifier is transferred maximally (resp. until ${}^{\diamond}\mathbb{N}$). As the double negation translation introduces a large number of negations, we will encounter formulas A for which $A \in \mathbb{T}$ is not explicitly defined by Definition 6, as BISH is intended as a 'negation-free' development of Mathematics ([2, p. 21]). Therefore, we list the following definition and Definition 20 below, implicit in Definition 6.

Definition 10 [$\mathbb{T}$] If A is $(\forall n \in {}^{\diamond}\mathbb{N})(\exists m \in {}^{*}\mathbb{N})\varphi(n,m)$, then $A \in \mathbb{T}$ is

$$A \to (\forall n \in {}^{*}\mathbb{N})(\exists m \in {}^{*}\mathbb{N})[m \text{ is } \max\{\diamond, n\}\text{-standard} \wedge \varphi(n,m)],$$

where $\max\{u,v\}$ is the maximum level of standardardness of u and v.

If A is '$(\exists n \in {}^{*}\mathbb{N})(\forall m \in {}^{*}\mathbb{N})[m \text{ is } \max\{\diamond, n\}\text{-standard} \to \varphi(n,m)]$', $A \in \mathbb{T}$ is

$$A \to (\exists n \in {}^{*}\mathbb{N})(\forall m \in {}^{*}\mathbb{N})[m \text{ is } \max\{*, n\}\text{-standard} \to \varphi(n,m)].$$

The previous formula is the first instance of 'Stratified Transfer' (See [6, 7]) in the context of $\mathbb{NSA}$. It would appear that the latter kind of Transfer Principle is essential to the study of the 'double negation shift' principle.

3.2 Non-constructive principles and $\mathcal{N}$

We first treat LPO, the *limited principle of omniscience*, which is essentially the law of excluded middle limited to Σ_1 formulas.

Theorem 11 *The system* $\mathbb{NSA}$ *proves* $\mathbb{LPO}^{\mathcal{N}}$.

Proof. By definition, $\mathbb{LPO}$ is $P \vee \sim P$ ($P \in \Sigma_1$). Then $(\mathbb{LPO})^{\mathcal{N}}$ is $(P \vee \sim P)^{\mathcal{N}}$ and

$$(P \vee \sim P)^{\mathcal{N}} \equiv \sim\big[\sim(P^{\mathcal{N}}) \wedge \sim(\sim P)^{\mathcal{N}}\big] \equiv \sim\big[\sim(\sim\sim P) \wedge \sim\sim(\sim\sim P)\big]. \tag{8}$$

Assuming P is $(\exists n \in \mathbb{N})\varphi(n, \boldsymbol{x})$, it is easy to see that $\sim P$ is $(\forall n \in \mathbb{N})\neg\varphi(n, \boldsymbol{x})$ and that $\sim\sim P$ is $(\exists n \in {}^{*}\mathbb{N})\varphi(n, \boldsymbol{x})$. Thus, (8) becomes

$$(\forall \boldsymbol{x} \in \mathbb{N}^k)\sim\big[\sim(\exists n \in {}^{*}\mathbb{N})\varphi(n, \boldsymbol{x}) \wedge \sim\sim(\exists n \in {}^{*}\mathbb{N})\varphi(n, \boldsymbol{x})\big], \tag{9}$$

and given that $\sim(\exists n \in {}^{*}\mathbb{N})\varphi(n, \boldsymbol{x})$ is $(\forall n \in {}^{\diamond}\mathbb{N})\neg\varphi(n, \boldsymbol{x})$ and $\sim(\forall n \in {}^{\diamond}\mathbb{N})\neg\varphi(n, \boldsymbol{x})$ is $(\exists n \in {}^{*}\mathbb{N})\varphi(n, \boldsymbol{x})$, then (9) becomes the following:

$$(\forall \boldsymbol{x} \in \mathbb{N}^k)\sim\big[(\forall n \in {}^{\diamond}\mathbb{N})\neg\varphi(n, \boldsymbol{x}) \wedge (\exists n \in {}^{*}\mathbb{N})\varphi(n, \boldsymbol{x})\big], \tag{10}$$

which is

$$(\forall \boldsymbol{x} \in \mathbb{N}^k)\Big[\big[(\forall n \in {}^{\diamond}\mathbb{N})\neg\varphi(n, \boldsymbol{x}) \wedge (\exists n \in {}^{*}\mathbb{N})\varphi(n, \boldsymbol{x})\big] \Rrightarrow 0 = 1\Big], \tag{11}$$

The formula in the small square brackets can be brought in existential or universal form. In the former, it is not suitable for Transfer, and in the latter it is. Hence, $(\forall n \in {}^{\diamond}\mathbb{N})\neg\varphi(n, \boldsymbol{x}) \wedge (\exists n \in {}^{*}\mathbb{N})\varphi(n, \boldsymbol{x})$ becomes

$$(\forall n \in {}^{\diamond}\mathbb{N})(\exists m \in {}^{*}\mathbb{N})[\neg\varphi(n, \boldsymbol{x}) \wedge \varphi(m, \boldsymbol{x})], \tag{12}$$

Now (12) and the fact that this formula is in $\mathbb{T}$ yields, by Definition 10, that

$$(\forall n \in {}^{*}\mathbb{N})(\exists m \in {}^{*}\mathbb{N})[m \text{ is } \max\{\diamond, n\}\text{-standard} \wedge \neg\varphi(n, \boldsymbol{x}) \wedge \varphi(m, \boldsymbol{x})],$$

Thus, (11) becomes that for all $\boldsymbol{x} \in \mathbb{N}^k$, we have

$$(\exists n \in {}^{*}\mathbb{N})(\forall m \in {}^{*}\mathbb{N})[m \text{ is } \max\{\diamond, n\}\text{-standard} \to \varphi(n, \boldsymbol{x}) \vee \neg\varphi(m, \boldsymbol{x})] \tag{13}$$

Now if $(\exists n_0 \in {}^{\diamond}\mathbb{N})\varphi(n_0, \boldsymbol{x}_0)$, then (13) holds for $\boldsymbol{x} = \boldsymbol{x}_0$. Now if $(\forall n_0 \in {}^{\diamond}\mathbb{N})\neg\varphi(n_0, \boldsymbol{x}_0)$, then (13) also holds for $\boldsymbol{x} = \boldsymbol{x}_0$ by putting $n = 0$ in (13). □

Hence, $(\mathbb{LPO})^{\mathcal{N}}$ is a tautology in $\mathbb{NSA}$. In particular, it follows from $P \vee \neg P$, the instance of LEM in $\mathbb{NSA}$ for ${}^{\diamond}\mathbb{N}$.

A weaker version of LPO is WLPO, which is $\neg\neg P \vee \neg P$ for $P \in \Sigma_1$. We now show that $\mathbb{NSA}$ proves $\mathbb{WLPO}^{\mathcal{N}}$, whereas $(\sim\mathbb{WLPO})^{\mathcal{N}}$ is a contradiction. This is not surprising, given that $\sim\mathbb{WLPO}$ is equivalent to a certain continuity principle (See [8, 10]) and that $\sim\mathbb{WLPO}$ contradicts $\mathbb{LPO}$.

Theorem 12 *The system* $\mathbb{NSA}$ *proves* $\mathbb{WLPO}^{\mathcal{N}}$ *and disproves* $(\sim\mathbb{WLPO})^{\mathcal{N}}$.

Proof. By definition, $(\mathbb{WLPO})^{\mathcal{N}}$ is $(\sim\sim P \mathbb{V} \sim P)^{\mathcal{N}}$ where $P \in \Sigma_1$, implying

$$(\sim\sim P \mathbb{V} \sim P)^{\mathcal{N}} \equiv \sim\big[\sim((\sim\sim P)^{\mathcal{N}}) \wedge \sim(\sim P)^{\mathcal{N}}\big] \equiv \sim\big[\sim(\sim\sim\sim\sim P) \wedge \sim\sim(\sim\sim P)\big]. \tag{14}$$

Assuming P is $(\exists n \in \mathbb{N})\varphi(n, \boldsymbol{x})$, $\sim\sim P$ is $(\exists n \in {}^*\mathbb{N})\varphi(n, \boldsymbol{x})$ and it is easy to see that the latter is also $\sim\sim\sim\sim P$. Thus, (14) becomes (10), given that $\sim(\exists n \in {}^*\mathbb{N})\varphi(n, \boldsymbol{x})$ is $(\forall n \in {}^{\diamond}\mathbb{N})\neg\varphi(n, \boldsymbol{x})$ and the first part of the theorem follows from Theorem 11. Furthermore, by definition, $(\sim\mathbb{WLPO})^{\mathcal{N}}$ is

$$\sim(\mathbb{WLPO}^{\mathcal{N}}) \equiv [\mathbb{WLPO}^{\mathcal{N}} \Rrightarrow 0 = 1] \equiv \neg(\mathbb{WLPO}^{\mathcal{N}}) \vee \neg[\mathbb{WLPO}^{\mathcal{N}} \in \mathbb{T}]. \tag{15}$$

By (13), $\mathbb{WLPO}^{\mathcal{N}}$ is a tautology and the first disjunct in (15) cannot hold. Furthermore, '$\mathbb{WLPO}^{\mathcal{N}} \in \mathbb{T}$' is the formula that, for all standard $\boldsymbol{x}$, (13) implies

$$(\exists n \in {}^*\mathbb{N})(\forall m \in {}^*\mathbb{N})[m \text{ is } \max\{*, n\}\text{-standard} \to \varphi(n, \boldsymbol{x}) \vee \neg\varphi(m, \boldsymbol{x})]. \tag{16}$$

Thus, $\neg[\mathbb{WLPO} \in \mathbb{T}]$ amounts to '(13) $\wedge$ $\neg$(16)'. We now show that (16) is a tautology in $\mathbb{NSA}$, implying that $(\sim[\mathbb{WLPO}])^{\mathcal{N}}$ is false. Now if $(\exists n_0 \in {}^*\mathbb{N})\varphi(n_0, \boldsymbol{x}_0)$, then (16) holds for $\boldsymbol{x} = \boldsymbol{x}_0$. Also, if $(\forall n_0 \in {}^*\mathbb{N})\neg\varphi(n_0, \boldsymbol{x}_0)$, then (16) also holds for $\boldsymbol{x} = \boldsymbol{x}_0$ by fixing n as any $n_1 \in {}^*\mathbb{N}$ in (16). □

We now consider LLPO, i.e. De Morgan's law $\neg(P \wedge Q) \to \neg P \vee \neg Q$ $(P, Q \in \Sigma_1)$.

Theorem 13 *The system* $\mathbb{NSA}$ *proves* $\mathbb{LLPO}^{\mathcal{N}}$ *and* $\mathbb{NIL}^{\mathcal{N}}$.

Proof. By definition, $\mathbb{LLPO}$ is $\sim(P \wedge Q) \Rrightarrow \sim P \mathbb{V} \sim Q$ for $P, Q \in \Sigma_1$. Hence, the antecedent of $\mathbb{LLPO}^{\mathcal{N}}$ is

$$\big[\sim(P \wedge Q)\big]^{\mathcal{N}} \equiv \sim(P^{\mathcal{N}} \wedge Q^{\mathcal{N}}) \equiv \sim(\sim\sim P \wedge \sim\sim Q),$$

whereas the consequent is

$$\big[\sim P \mathbb{V} \sim Q\big]^{\mathcal{N}} \equiv \sim\big(\sim(\sim P)^{\mathcal{N}} \wedge \sim(\sim Q)^{\mathcal{N}}\big) \equiv \sim(\sim\sim\sim\sim P \wedge \sim\sim\sim\sim Q).$$

By the above $\sim\sim R$ and $\sim\sim\sim\sim R$ are identical for $R \in \Sigma_1$ and the first case is done. For $\mathbb{NIL}^{\mathcal{N}}$, the antecedent expresses that $xy = 0$ with $\mathbb{N}$ replaced with ${}^*\mathbb{N}$. This implies that $x = 0$ or $y = 0$ with $\mathbb{N}$ replaced by ${}^*\mathbb{N}$. However, the consequent of $\mathbb{NIL}^{\mathcal{N}}$, i.e. $\sim(\sim(x = 0) \wedge \sim(y = 0))$ exactly expresses that $x = 0$ or $y = 0$ with $\mathbb{N}$ replaced by ${}^*\mathbb{N}$; Hence we are done. □

Next, we consider Markov's principle MP, which is essentially double negation elimination, i.e. $\neg\neg P \to P$, $(P \in \Sigma_1)$. A weak version of MP is $\mathrm{MP}^\vee$, i.e. $\neg(\neg P \wedge \neg Q) \to \neg\neg P \vee \neg\neg Q$. Finally, Weak Markov's principle WMP is $(\forall x \in \mathbb{R})\big[(\forall y \in \mathbb{R})(\neg\neg(y > 0) \vee \neg\neg(y < x)) \to x > 0\big]$.

Theorem 14 *The system* $\mathbb{NSA}$ *can prove* $\mathbb{MP}^{\mathcal{N}}$, $\mathbb{WMP}^{\mathcal{N}}$ *and* $(\mathbb{MP}^\vee)^{\mathcal{N}}$.

Proof. Given the absence of disjunction, $\sim\sim P \Rrightarrow P$ $(P \in \Sigma_1)$ is quite easy to translate. Indeed, we have

$$\big[\sim\sim P \Rrightarrow P\big]^{\mathcal{N}} \equiv (\sim\sim P)^{\mathcal{N}} \Rrightarrow P^{\mathcal{N}} \equiv \sim\sim\sim\sim P \Rrightarrow \sim\sim P. \tag{17}$$

Assuming P is $(\exists n \in \mathbb{N})\varphi(n, \boldsymbol{x})$, it is easy to see that $\sim\sim P$ is $(\exists n \in {}^*\mathbb{N})\varphi(n, \boldsymbol{x})$. Furthermore, $\sim\sim(\exists n \in {}^*\mathbb{N})\varphi(n, \boldsymbol{x})$ is $(\exists n \in {}^*\mathbb{N})\varphi(n, \boldsymbol{x})$, and (17) becomes

$$(\exists n \in {}^*\mathbb{N})\varphi(n, \boldsymbol{x}) \Rrightarrow (\exists n \in {}^*\mathbb{N})\varphi(n, \boldsymbol{x}) \equiv (\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x}) \to (\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x}),$$

which is another tautology regarding ${}^\diamond\mathbb{N}$.

For $\mathbb{WMP}^{\mathcal{N}}$, we consider its antecedent

$$[\sim\sim(0 < y) \,\mathbb{V}\, \sim\sim(y < x)]^{\mathcal{N}} \equiv \sim[\sim\sim\sim(0 < y)^{\mathcal{N}} \wedge\ \sim\sim\sim(y < x)^{\mathcal{N}}],$$

where the latter is $\sim[\sim\sim\sim\sim\sim(0 < y) \wedge \sim\sim\sim\sim\sim(y < x)]$. Then $\mathbb{WMP}^{\mathcal{N}}$ is

$$(\forall x \in \mathbb{R})\big(\big[(\forall y \in \mathbb{R})\sim[\sim\sim\sim\sim\sim(0 < y) \wedge \sim\sim\sim\sim\sim(y < x)]\big] \Rrightarrow \sim\sim(x > 0)\big).$$

Applying the antecedent for $y = 0$, we obtain $(\exists n \in {}^*\mathbb{N})(q_n > \frac{1}{2^n})$, if x is $(q_n)_{n\in\mathbb{N}}$. But this is the consequent $\sim\sim(x > 0)$ and this case is done.

For $(\mathbb{MP}^\vee)^{\mathcal{N}}$, the antecedent is $\big[\sim(\sim P \wedge \sim Q)\big]^{\mathcal{N}}$, which is $\sim(\sim\sim\sim P \wedge \sim\sim\sim Q)$, by definition. Similarly, the antecedent is

$$\big[\sim\sim P \,\mathbb{V}\, \sim\sim Q\big]^{\mathcal{N}} \equiv \sim\big[\sim(\sim\sim P^{\mathcal{N}}) \wedge \sim(\sim\sim Q^{\mathcal{N}}) \equiv \sim(\sim\sim\sim\sim\sim P \wedge \sim\sim\sim\sim\sim Q).$$

Hence we are done, as $\sim\sim\sim R$ and $\sim\sim\sim\sim\sim R$ are identical for $R \in \Sigma_1$. □

Remark 15 The above results suggest that $\mathcal{N}$ produces statements about properties of ${}^\diamond\mathbb{N}$ in $\mathbb{NSA}$. Furthermore, it is easy verify that, $\mathbb{FAN}^{\mathcal{N}}$ (resp. $\mathbb{WKL}^{\mathcal{N}}$) is essentially $\mathbb{FAN}$ (resp. $\mathbb{WKL}$), but with each instance of u-standard replaced by $\max\{\diamond, u\}$-standard, i.e. $\mathbb{FAN}^{\mathcal{N}}$ is $\mathbb{FAN}$ *relative to* ${}^\diamond\mathbb{N}$. This suggests that $\mathbb{NSA}$ should be expanded, perhaps via the law of non-contradiction, with axioms regarding ${}^\diamond\mathbb{N}$ to make $\mathcal{N}$ work fully.

3.3 An alternative to the double negation translation $\mathcal{N}$

In this section, we discuss $\mathcal{M}$, an alternative to $\mathcal{N}$ which makes use of the nonstandard nature of $\mathbb{NSA}$. In the BHK-interpretation, the meaning of $\vee$ and $\exists$ is much stronger than the 'classical' meaning. From Definition 2, it is clear that

$\mathcal{N}$ replaces these 'essentially constructive' objects with a weaker version. Exactly the same thing happens for $\mathbb{V}$ and $\exists$ for $\mathcal{N}$ in $\mathbb{NSA}$, as is clear from Definition 9. However, there are alternative *extremely obvious* weakenings of $\mathbb{V}$ and $\exists$ in $\mathbb{NSA}$ already, namely $\vee$ for $\mathbb{V}$ and $(\exists x \in {}^\diamond X)$ for $(\exists x \in X)$. Thus, define $\mathcal{M}$ in the same way as $\mathcal{N}$ in Definition 9, but with the following[8] changes:

$$(A \mathbb{V} B)^{\mathcal{M}} \equiv A^{\mathcal{M}} \vee B^{\mathcal{M}} \text{ and } \big[(\exists x \in X)A(x)\big]^{\mathcal{M}} \equiv (\exists x \in {}^\diamond X)[A(x)]^{\mathcal{M}}. \quad (18)$$

The interpretation $\mathcal{M}$ is motivated by Remark 15 and the reader can verify that ${}^\diamond\mathbb{N}$ cannot be replaced by ${}^*\mathbb{N}$ in (18). We now briefly study the behavior of $\mathcal{M}$ on non-constructive principles.

Theorem 16 *The system* $\mathbb{NSA}$ *proves* $\mathbb{LPO}^{\mathcal{M}}$, $\mathbb{WLPO}^{\mathcal{M}}$, $\mathbb{NIL}^{\mathcal{M}}$, *and* $\mathbb{LLPO}^{\mathcal{M}}$.

Proof. For $\mathbb{LPO}^{\mathcal{M}}$, assume $P \in \Sigma_1$ is $(\exists n \in \mathbb{N})\varphi(n, \boldsymbol{x})$ and consider the following:

$$(P \mathbb{V} \sim P)^{\mathcal{M}} \equiv P^{\mathcal{M}} \vee \sim(P^{\mathcal{M}}) \equiv (\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x}) \vee \sim\big[(\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x})\big].$$

As $\sim[(\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x})]$ is $(\forall n \in {}^\diamond\mathbb{N})\neg\varphi(n, \boldsymbol{x})$, the first case is done. For the second case, i.e. for $\mathbb{WLPO}^{\mathcal{M}}$, consider the following:

$$(\sim\sim P \mathbb{V} \sim P)^{\mathcal{M}} \equiv \sim\sim P^{\mathcal{M}} \vee \sim(P^{\mathcal{M}}) \equiv \sim\sim(\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x}) \vee \big[(\forall n \in {}^\diamond\mathbb{N})\neg\varphi(n, \boldsymbol{x})\big].$$

As $\sim\sim[(\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x})]$ is $(\exists n \in {}^*\mathbb{N})\varphi(n, \boldsymbol{x})$, the second case is also trivial.

For $\mathbb{LLPO}^{\mathcal{M}}$, it is easy to verify that both the consequent and antecedent are $(\forall n_1 \in {}^*\mathbb{N})\neg\varphi_1(n_2, \boldsymbol{x}) \vee (\forall n_2 \in {}^*\mathbb{N})\neg\varphi_2(n_2, \boldsymbol{x})$ after resolving the hyperimplication, where P, Q are $(\exists n_i \in {}^*\mathbb{N})\varphi_i(n_i, \boldsymbol{x})$ for $i = 1, 2$. For $\mathbb{NIL}^{\mathcal{M}}$, the antecedent expresses that $xy = 0$ with $\mathbb{N}$ replaced with ${}^*\mathbb{N}$. This implies that $x = 0$ or $y = 0$ with $\mathbb{N}$ replaced by ${}^*\mathbb{N}$. However, the consequent of $\mathbb{NIL}^{\mathcal{M}}$, i.e. $x = 0 \vee y = 0$ and the fact that this formula is in $\mathbb{T}$ exactly expresses that $x = 0$ or $y = 0$ with $\mathbb{N}$ replaced by ${}^*\mathbb{N}$; Hence we are done. □

Theorem 17 *The system* $\mathbb{NSA}$ *proves* $\mathbb{MP}^{\mathcal{M}}$, $\mathbb{WMP}^{\mathcal{M}}$, *and* $(\mathbb{MP}^\vee)^{\mathcal{M}}$.

Proof. For $\mathbb{MP}^{\mathcal{M}}$, assume $P \in \Sigma_1$ is $(\exists n \in \mathbb{N})\varphi(n, \boldsymbol{x})$ and consider the following:

$$(\sim\sim P \Rrightarrow P)^{\mathcal{M}} \equiv \sim\sim P^{\mathcal{M}} \Rrightarrow P^{\mathcal{M}} \equiv \sim\sim(\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x}) \Rrightarrow (\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x}).$$

Now $\sim\sim(\exists n \in {}^\diamond\mathbb{N})\varphi(n, \boldsymbol{x})$ is $(\exists n \in {}^*\mathbb{N})\varphi(n, \boldsymbol{x})$ and resolving the hyperimplication in the pervious formula yields a triviality, proving the first case. For the second case, the antecedent of $\mathbb{WMP}^{\mathcal{M}}$ is $\sim\sim(y > 0)^{\mathcal{M}} \vee \sim\sim(x > y)^{\mathcal{M}}$, for all $y \in \mathbb{R}$. In particular, for $y = 0$, we have $\sim\sim(x > 0)^{\mathcal{M}}$. Now if x is $(q_n)_{n\in\mathbb{N}}$, then $(x > 0)^{\mathcal{M}}$ is $(\exists n \in {}^\diamond\mathbb{N})(q_n > \frac{1}{2^n})$ and $\sim\sim(x > 0)^{\mathcal{M}}$ is thus $(\exists n \in {}^*\mathbb{N})(q_n > \frac{1}{2^n})$. Thus, in the same way as for $\mathbb{MP}^{\mathcal{M}}$, we observe that $\mathbb{WMP}^{\mathcal{M}}$ is provable in $\mathbb{NSA}$. Analogously, for $(\mathbb{MP}^\vee)^{\mathcal{M}}$, both the antecedent and consequent are $(\exists n_1 \in {}^*\mathbb{N})\varphi_1(n_2, \boldsymbol{x}) \vee (\exists n_2 \in {}^*\mathbb{N})\varphi_2(n_2, \boldsymbol{x})$, where P, Q are $(\exists n_i \in {}^*\mathbb{N})\varphi_i(n_i, \boldsymbol{x})$ for $i = 1, 2$. □

[8] For the second case, it may be necessary to add $\diamond$ to the universal formulas in A.

4 Bibliography

[1] *Handbook of mathematical logic*, North-Holland Publishing Co., Amsterdam, 1977. Edited by Jon Barwise; With the cooperation of H. J. Keisler, K. Kunen, Y. N. Moschovakis and A. S. Troelstra; Studies in Logic and the Foundations of Mathematics, Vol. 90.

[2] Errett Bishop, *Foundations of constructive analysis*, McGraw-Hill Book Co., New York, 1967.

[3] ———, *Foundations of constructive analysis*, Ishi Press International, Tokyo, 2012. Reprint of [2] with a new foreword by Michael Beeson.

[4] Douglas S. Bridges, *Constructive mathematics: a foundation for computable analysis*, Theoret. Comput. Sci. **219** (1999), no. 1-2, 95–109.

[5] Samuel R. Buss, *An introduction to proof theory*, Handbook of proof theory, Stud. Logic Found. Math., vol. 137, North-Holland, Amsterdam, 1998, pp. 1–78.

[6] Karel Hrbacek, *Relative Set Theory: Internal View*, J. Log. Anal. **1** (2009), Paper 8, pp. 108.

[7] ———, *Relative Set Theory: Some external issues*, J. Log. Anal. **2** (2010), Paper 8, pp. 37.

[8] Hajime Ishihara, *Reverse mathematics in Bishop's constructive mathematics*, Philosophia Scientiae (Cahier Spécial) **6** (2006), 43-59.

[9] ———, *Constructive reverse mathematics: compactness properties*, From sets and types to topology and analysis, Oxford Logic Guides, vol. 48, 2005.

[10] Sam Sanders, *Reuniting the antipodes: Bringing together Nonstandard Analysis and Constructive Analysis*, Submitted (2012), pp. 49.

[11] ———, *Reverse engineering Constructive Reverse Mathematics*, Submitted (2012).

[12] ———, *Algorithm and Proof as Ω-invariance and Transfer: A new model of computation in Nonstandard Analysis*, Electronic Proceedings in Computer Science, DCM (2012), In Press.

[13] ———, *On the connection between Nonstandard Analysis and Constructive Analysis*, To appear in Logique et Analyse (2013), pp. 19.

[14] ———, *On algorithm and robustness in a Non-standard sense* (Hanne Andersen, Dennis Dieks, Wenceslao Gonzalez, Thomas Übel, and Gregory Wheeler, eds.), The Philosophy of Science in a European Perspective, Springer, 2013.

[15] Peter Schuster, Ulrich Berger, and Horst Osswald (eds.), *Reuniting the antipodes: constructive and nonstandard views of the continuum*, Synthese Library, vol. 306, Kluwer, 2001.

[16] Anne Sjerp Troelstra, *Metamathematical investigation of intuitionistic arithmetic and analysis*, Springer Berlin, 1973. Lecture Notes in Mathematics, Vol. 344.

[17] Keita Yokoyama, *Standard and non-standard analysis in second order arithmetic*, Tohoku Mathematical Publications, vol. 34, Sendai, 2009. PhD Thesis, Tohoku University, 2007.

[18] ———, *Formalizing non-standard arguments in second-order arithmetic*, J. Symbolic Logic **75** (2010), no. 4, 1199–1210.

A Appendix: The system $\mathbb{NSA}$

In this section, we introduce the system $\mathbb{NSA}$. We adopt Keita Yokoyama's approach to Nonstandard Analysis from [17, 18].

The language $\mathcal{L}_2$ of second-order arithmetic has two sorts of variables: number variables x, y, z, etc, and set variables X, Y, Z, etc. The language $\mathcal{L}_2^*$ of *nonstandard* second-order arithmetic has four sorts of variables: standard number variables x^s, y^s, z^s, etc, standard set variables X^s, Y^s, Z^s, etc, nonstandard number variables x^*, y^*, z^*, etc, and nonstandard set variables X^*, Y^*, Z^*, etc. The range of these variables is $V^s = (M^s, S^s)$ and $V^* = (M^*, S^*)$, where the first (resp. second) entry is for number (resp. set) variables. The usual symbols 0, 1, $=$, $+$, $\times$, $<$ and $\in$ are present (for both the standard and nonstandard language), plus a new symbol '$\sqrt{}$' which serves as an embedding of the standard into the nonstandard universe.

With regard to notation, we abbreviate $\sqrt{x^s}$ and $\sqrt{X^s}$ as $\hat{x}^s$ and $\hat{X}^s$. For a sentence $\Phi \in \mathcal{L}_2$, Φ^s (resp. Φ^*) is obtained by appending 's' (resp. '$*$') to all variables in Φ. Furthermore, we will denote the usual symbols 0, 1, $=$, $+$, $\times$, $<$ and $\in$, *without* such suffixes, as it will always be clear whether the context is the (non)standard universe. Finally, if we have $(\forall y^s)(\hat{y}^s < x^*)$, we write '$x^* \in \Omega$' and say that '$x^*$ is infinite'. A number is 'finite' if it is not infinite.

We now define s-$\mathbb{NSA}$, a nonstandard version of RCA_0 where recursive comprehension is replaced by Ω-CA, i.e. comprehension for Ω-invariant formulas.

Definition 18 (s-$\mathbb{NSA}$)

1. **Induction**: $I\Sigma_1^s$ and $I\Sigma_1^*$.
2. **Nonstandard Universe**: "$\sqrt{} : V^s \to V^*$ is an injective homomorphism".
3. **End extension:**

$$(\forall x^*, y^s)\big[(x^* < \hat{y}^s) \to (\exists z^s)(x^* = z^s)\big].$$

4. **Transfer**: For any $\varphi \in \Delta_0$, we have

$$(\forall x^s, X^s)\big[\varphi(x^s, X^s)^s \leftrightarrow \varphi(\hat{x}^s, \hat{X}^s)^*\big].$$

5. **Ω-CA**: For all $\varphi \in \Delta_0$, we have

$$\begin{aligned}(\forall x^s)(\forall y^*, z^* \in \Omega)\big[\varphi(\hat{x}^s, y^*)^* \leftrightarrow \varphi(\hat{x}^s, z^*)^*\big] \\ \to (\exists Z^s)(\forall w^s)\big(\forall v^* \in \Omega)(w^s \in Z^s \leftrightarrow \varphi(\hat{z}^s, v^*)^*\big).\end{aligned}$$

It is not difficult to show that s-$\mathbb{NSA}$ + Π_1-TRANS is a conservative extension of ACA_0 using the results from [18]. In a forthcoming paper, we also show that s-$\mathbb{NSA}$ is a conservative extension of RCA_0. Thus, Turing computability corresponds to Ω-invariance. Hence, s-$\mathbb{NSA}$ cannot capture BISH properly. For this reason, we extend the language $\mathcal{L}_2^*$ with two more sorts: $x^\diamond, y^\diamond, z^\diamond$, etc. and $X^\diamond, Y^\diamond, Z^\diamond$, etc; The domain for these variables being $V^\diamond = (M^\diamond, S^\diamond)$. Two new symbols $\sqrt[1]{}$ and $\sqrt[2]{}$ will replace the original embedding $\sqrt{}$ from s-$\mathbb{NSA}$. For a formula $\Phi \in \mathcal{L}_2$, define $\Phi^\diamond$ as in the same way as Φ^s and Φ^*. We abbreviate $\sqrt[1]{x^s}$ and $\sqrt[1]{X^s}$ as $\hat{x}^s$ and $\hat{X}^s$, $\sqrt[2]{x^\diamond}$ and $\sqrt[2]{X^\diamond}$ as $\hat{x}^\diamond$ and $\hat{X}^\diamond$, and $\sqrt[2]{\sqrt[1]{x^s}}$ and $\sqrt[2]{\sqrt[1]{X^s}}$

as $\hat{x}^s$ and $\hat{\hat{X}}^s$. The set Ω is defined as above. Intuitively, $V^\diamond$ is an 'intermediate' nonstandard universe between V^s and V^*, and the Transfer principle of $\mathbb{NSA}$ transfers suitable bounded formulas between these universes.

Secondly, to accommodate statements of the form 'There exists a discontinuous function $f : 2^{\mathbb{N}} \to \mathbb{N}$', we need *sets of sets*. Thus, let $\mathfrak{X}^s, \mathfrak{Y}^s, \mathfrak{Z}^s$, etc denote variables of a new sort called *standard sets of sets*, and extend V^s to (M^s, S^s, U^s), where U^s is the range of the new variables. We assume that $V^\diamond$ and V^* are extended analogously. Finally, we assume that functions are defined from sets as usual.

Definition 19 ($\mathbb{NSA}$)

1. **Induction**: $I\Sigma_1^s$, $I\Sigma_1^\diamond$ and $I\Sigma_1^*$.
2. **Nonstandard Universe**$^\diamond$: "The mapping $\surd_1 : V^s \to V^\diamond$ is an injective homomorphism".
3. **Nonstandard Universe***: "The mapping $\surd_2 : V^\diamond \to V^*$ is an injective homomorphism".
4. **End extension**$^\diamond$: $M^\diamond$ is an end extension of M^s.
5. **End extension***: M^* is an end extension of $M^\diamond$.
6. **Transfer**$^\diamond$: For any $\varphi \in \Delta_0$, we have
$$(\forall x^s, X^s)\big[\varphi(x^s, X^s)^s \leftrightarrow \varphi(\hat{x}^s, \hat{X}^s)^\diamond \leftrightarrow \varphi(\hat{\hat{x}}^s, \hat{\hat{X}}^s)^*\big].$$
7. **Transfer***: For any $\varphi \in \Delta_0$, we have
$$(\forall x^\diamond, X^\diamond)\big[\varphi(x^\diamond, X^\diamond)^\diamond \leftrightarrow \varphi(\hat{x}^\diamond, \hat{X}^\diamond)^*\big].$$
8. **Ω-CA**: For all $\mathfrak{F}^s : M^s \times S^s \times M^s \to M^s$, we have
$$(\forall x^s, X^s)(\forall y^*, z^* \in \Omega)\big[\hat{\hat{\mathfrak{F}}}^s(\hat{\hat{x}}^s, \hat{\hat{X}}^s, y^*)^* = \hat{\hat{\mathfrak{F}}}^s(\hat{\hat{x}}^s, \hat{\hat{X}}^s, z^*)^*\big]$$
$$\to (\exists \mathfrak{G}^s : M^s \times S^s \to M^s)(\forall w^s, W^s)(\forall v^* \in \Omega)(\hat{\hat{\mathfrak{G}}}^s(w^s, W^s) = \hat{\hat{\mathfrak{F}}}^s(\hat{\hat{w}}^s, , \hat{\hat{W}}^s, v^*)^*).$$
9. **Countable Choice**: For all decidable φ
$$(\forall n \in \mathbb{N})(\exists x \in X)\varphi(n, x) \Rrightarrow (\exists h \in \mathbb{N}^X)(\forall n \in \mathbb{N})\varphi(n, h(n)).$$
10. **Transfer Rule** For $A \in \Pi_1^0$, we have $\frac{A}{A \wedge (A \in \mathbb{T})}$.

B Appendix: Technical Results

In this section, we prove the items 3 and 7 of Definition 9. We did not consider these statements in [10, 12], as BISH is intended to be a 'negation-free' (or 'positive' or 'affirmative') part of Mathematics by [2, p. 21]. To this end, we require some small refinements to $\mathbb{B}$, starting with the following extension of $\mathbb{T}$.

Definition 20 [$\mathbb{T}$]

1. If $A \vee B$ is unsuitable for Transfer, $(A \vee B) \in \mathbb{T}$ is defined as

$$A \vee B \to \big[(A \wedge A \in \mathbb{T}) \vee (B \wedge B \in \mathbb{T})\big]. \tag{19}$$

2. The formula '$(A \in \mathbb{VT}) \in \mathbb{T}$' is '$0 = 0$' and '$[\neg(A \in \mathbb{VT})] \in \mathbb{T}$' is '$0 = 1$'.

The motivation behind the first item is as follows: Even if $A \vee B$ is unsuitable for Transfer, it is possible that one of the formulas A or B *is* suitable for Transfer. For instance, if A is suitable for Transfer, but B and $A \vee B$ are not, then (19) is

$$A \vee B \to \big[(A \wedge A \in \mathbb{T}) \vee B\big], \tag{20}$$

as $B \in \mathbb{T}$ is $0 = 0$. Together with $A \vee B$, (20) becomes $(A \wedge A \in \mathbb{T}) \vee B$, as one intuitively expects to derive from $(A \vee B) \wedge [(A \vee B) \in \mathbb{T}]$, in this case. Furthermore, if A and B are both unsuitable for Transfer, then (19) is trivial.

The motivation behind the second item is as follows: By Theorem 8, Π_1-TRANS is equivalent to $\mathbb{LPO}$. Now it is not difficult to show that $\mathbb{LPO} \in \mathbb{T}$ is trivial in $\mathbb{NSA}$, and the same holds for instances of $\mathbb{LPO}$. However, if Φ is an instance of $\neg\mathbb{LPO}$, then $\Phi \in \mathbb{T}$ is false in $\mathbb{NSA}$. By analogy, $(A \in \mathbb{VT}) \in \mathbb{T}$ should be trivial and $[\neg(A \in \mathbb{VT})] \in \mathbb{T}$ should be false.

Clearly, the first item of Definition 20 expresses that '$\mathbb{T}$ should be maximal', while the second item expresses that we cannot learn more from $A \in \mathbb{VT}$ via Transfer. As $A \in \mathbb{VT}$ is an instance of Disjunctive Transfer, this not surprising.

Furthermore, in Theorem 9, $A \Vbar B$ occurs negatively, i.e. in the scope of a hypernegation, a situation not usually encountered in Constructive Analysis. In line with Definition 5, we have the following definition.

Definition 21 [Hyperdisjunction] If $A(\boldsymbol{x}) \Vbar B(\boldsymbol{x})$ occurs positively in $(\forall \boldsymbol{x} \in \mathbb{N}^k)\Phi(x)$, then the latter[9] is $(\exists \psi \in \Delta_0)\big[\psi(\boldsymbol{x}, \omega)$ is Ω-invariant $\wedge (\forall \boldsymbol{x} \in \mathbb{N}^k)\widetilde{\Phi(x)}\big]$, where $\widetilde{\Phi(\boldsymbol{x})}$ is $\Phi(\boldsymbol{x})$ with $A(\boldsymbol{x}) \Vbar B(\boldsymbol{x})$ replaced by (3).

If $A(\boldsymbol{x}) \Vbar B(\boldsymbol{x})$ occurs negatively in $(\forall \boldsymbol{x} \in \mathbb{N}^k)\Phi(x)$, the latter is $(\forall \psi \in \Delta_0)\big[\psi(\boldsymbol{x}, \omega)$ is Ω-invariant $\to (\forall \boldsymbol{x} \in \mathbb{N}^k)\widetilde{\Phi(x)}$. If occurrences are in both categories, the extra quantifiers involving ψ yield an additional Σ_2-prefix.

Theorem 22 *In* $\mathbb{NSA}$*, formulas* A, B *satisfy*

$$\sim(A \Vbar B) \iff \sim A \wedge \sim B \text{ and } (\forall x) \sim A(x) \iff \sim\big[(\exists x) A(x)\big]. \tag{21}$$

Proof. We first prove the first claim. Let $A(\boldsymbol{x})$ and $B(\boldsymbol{x})$ be as stated with all standard free variables $\boldsymbol{x}$ as shown. We need to prove that

$$(\forall \boldsymbol{x} \in \mathbb{N}^k)\big[\sim(A(\boldsymbol{x}) \Vbar B(\boldsymbol{x})) \iff (\sim A(\boldsymbol{x}) \wedge \sim B(\boldsymbol{x}))\big]. \tag{22}$$

[9] Note that the formula '$\psi(\boldsymbol{x}, \omega)$ is Ω-invariant' has no free variables $\boldsymbol{x}$.

The right-hand side of (22) is $\neg[A(\boldsymbol{x}) \wedge A(\boldsymbol{x}) \in \mathbb{T}] \wedge \neg[B(\boldsymbol{x})\mathbb{B}(\boldsymbol{x}) \in \mathbb{T}]$ and the left-hand side of (22) can be brought in the following form, in both cases distinguished by Definition 21, using only the classical logic of $\mathbb{NSA}$.

$$(\forall \psi \in \Delta_0)\big(\psi(\boldsymbol{x},\omega) \text{ is } \Omega\text{-invariant} \rightarrow \sim\big([\psi(\boldsymbol{x},\omega) \rightarrow [A(\boldsymbol{x}) \wedge A(\boldsymbol{x}) \in \mathbb{T}]] \wedge [\neg\psi(\boldsymbol{x},\omega) \rightarrow [B(\boldsymbol{x}) \wedge B(\boldsymbol{x}) \in \mathbb{T}]]\big)\big).$$

As the formula inside the hypernegation is not suitable for Transfer, the hypernegation reduces to negation, yielding

$$(\forall \psi \in \Delta_0)\big(\psi(\boldsymbol{x},\omega) \text{ is } \Omega\text{-invariant} \rightarrow [\psi(\boldsymbol{x},\omega) \wedge \neg[A(\boldsymbol{x}) \wedge A(\boldsymbol{x}) \in \mathbb{T}]] \vee [\neg\psi(\boldsymbol{x},\omega) \wedge \neg[B(\boldsymbol{x}) \wedge B(\boldsymbol{x}) \in \mathbb{T}]]\big). \quad (23)$$

Applying the previous formula for a true and a false instance of Ω-invariant $\psi(\boldsymbol{x},\omega)$, we obtain that $\neg[A(\boldsymbol{x})\wedge A(\boldsymbol{x}) \in \mathbb{T}]$ and $\neg[B(\boldsymbol{x})\wedge B(\boldsymbol{x}) \in \mathbb{T}]$, i.e. $\sim A\wedge\sim B$. Similarly, if the latter formula holds, then (23) is true by classical logic.

For the second claim, note that $(\forall x)\sim A(x)$ is $(\forall x)[A(x) \Rrightarrow 0 = 1]$, yielding

$$(\forall x)\big[(A(x) \wedge A(x) \in \mathbb{T}) \rightarrow 0 = 1\big] \equiv (\forall x)\neg[A(x) \wedge A(x) \in \mathbb{T}].$$

Now $\sim\big[(\exists x)A(x)\big]$ is $(\exists x)A(x) \Rrightarrow 0 = 1$, implying

$$\big[(\exists x)A(x)\wedge[(\exists x)A(x)] \in \mathbb{T}\big] \rightarrow 0 = 1, \text{ which is } \neg\big[A \in \mathbb{V}\mathbb{T}\wedge(\exists x)[A(x)\wedge A(x) \in \mathbb{T}]\big],$$

by Definition 6. The previous formula yields $\neg(A \in \mathbb{V}\mathbb{T})\vee(\forall x)\neg[A(x)\wedge A(x) \in \mathbb{T}]$, which is $\neg(A \in \mathbb{V}\mathbb{T}) \vee (\forall x)\sim A(x)$. Thus, we immediately obtain $(\forall x)\sim A(x) \Rrightarrow \sim\big[(\exists x)A(x)\big]$, by Definition 20, i.e. the forward direction in the second item of (21). For the other direction, i.e. $\sim\big[(\exists x)A(x)\big] \Rrightarrow (\forall x)\sim A(x)$, note that the antecedent of the latter is

$$\big[\neg(A \in \mathbb{V}\mathbb{T}) \vee (\forall x)\sim A(x)\big] \wedge \Big[\big[\neg(A \in \mathbb{V}\mathbb{T}) \vee (\forall x)\sim A(x)\big] \in \mathbb{T}\Big].$$

Now the second part of this conjunction is the implication $C \rightarrow D$, where C is $\neg(A \in \mathbb{V}\mathbb{T}) \vee (\forall x)\sim A(x)$ and where D is

$$\big[\neg(A \in \mathbb{V}\mathbb{T}) \wedge [\neg(A \in \mathbb{V}\mathbb{T})] \in \mathbb{T}\big] \vee \big[(\forall x)\sim A(x) \wedge \big([(\forall x)\sim A(x)] \in \mathbb{T}\big)\big]$$

Again by Definition 20, the left-hand side of this disjunction is false. Thus, we must have the right-hand side of the previous formula, which means that $\sim\big[(\exists x)A(x)\big] \Rrightarrow (\forall x)\sim A(x)$ is immediate. □

Decidability and the multiplicative structure of positive rationals together with the 'greatest common divisor'

Alla Sirokofskich*

[1] Department of Mathematics
University of Crete, 714 09 Heraklion, Greece
asirokof@math.uoc.gr
[2] School of Mathematics
University of Leeds, Leeds, UK
pmtas@leeds.ac.uk

Abstract. We consider the ring $\mathbb{Z}^\infty$ of integer sequences, under component-wise addition, or, equivalently, the multiplicative group $\mathbb{Q}^+$ of positive rationals, with the extra structure, which, in $\mathbb{Q}^+$, corresponds to the 'greatest common divisor'. We show that an extension of this structure has a decidable theory.

1 Introduction

In what follows $\mathbb{Z}$ is the ring of integers. By $\mathbb{Z}^\infty$ we denote the set of all functions from $\mathbb{Z}$ to $\mathbb{Z}$ with finite support; that is, we consider only the functions $x : \mathbb{Z} \to \mathbb{Z}$ for which the set $\{i \in \mathbb{Z} : x(i) \neq 0\}$ is finite. We use the notation $(x)_i = x(i)$. We regard the elements of $\mathbb{Z}^\infty$ as infinite sequences, (infinite in both directions), with only a finite set of non-zero coordinates. It is worth mentioning that we could as well consider the set of all functions from $\mathbb{N}$ to $\mathbb{Z}$ since this will not affect the results of this paper. By $\mathbb{N}$ we denote the set of positive integers. By $\mathbb{N}_0$ we denote the set of non-negative integers.

Observe that the additive group $(\mathbb{Z}^\infty, +)$ is isomorphic to the multiplicative group $(\mathbb{Q}^+, \times)$, where $\mathbb{Q}^+$ is the set of positive rationals. Indeed, let the sequence $(p_i)_{i\in\mathbb{N}}$ be the natural enumeration of the set of primes. Then the element $n = \prod p_i^{m_i}$ of $\mathbb{Q}^+$ corresponds to the element $x \in \mathbb{Z}^\infty$ with $(x)_i = m_i$ for $i \in \mathbb{Z}$. We can define a notion of divisibility $|$ in $\mathbb{Q}^+$ as follows: let $n_1 = \prod p_i^{m_i}$ and $n_2 = \prod p_i^{k_i}$, then

$$n_1|n_2 \iff \forall i(m_i \leq k_i).$$

Thus the restriction of the *min* function of the structure $\mathcal{A}$ on the set $A^+ = \{x \in \mathbb{Z}^\infty : \forall i \in \mathbb{Z}[(x)_i \geq 0]\}$ is interpreted as the greatest common divisor (gcd) in the substructure $(\mathbb{Z}^+, \times, =)$ of $(\mathbb{Q}^+, \times, =)$, where, with the notation as above,

$$gcd(n_1, n_2) = \prod p_i^{min\{m_i,k_i\}}.$$

We extend this definition for n_1, n_2 in $\mathbb{Q}^+$. It is easy to see that the gcd is definable from $|$ in the substructure $(\mathbb{Z}^+, |, =)$ of $(\mathbb{Q}^+, |, =)$. Indeed, for $x, y \in \mathbb{Z}$ we have that

$$d = gcd(x, y) \text{ if and only if } (d|x) \wedge (d|y) \wedge \forall w[w|x \wedge w|y \to w|d]. \tag{1}$$

Note that $|$ is existentially definable in $(\mathbb{Z}^+, \times, =)$ by

$$x|y \text{ if and only if } \exists z[x \times z = y]. \tag{2}$$

*The research project is implemented within the framework of the Action Supporting Postdoctoral Researchers of the Operational Program "Education and Lifelong Learning" (Actions Beneficiary: General Secretariat for Research and Technology), and is co-financed by the European Social Fund (ESF) and the Greek State.

The relation (2) does not extend to the multiplicative rationals. Moreover, the 'gcd' is not definable in $(\mathbb{Q}^+, \times, =)$, because of the following argument: Any module over a fixed ring has stable theory in the language of modules, (see [4]). On the other hand, the notion of 'gcd' implies some kind of ordering, namely the formula $\varphi(x,y)$: 'gcd'$(x,y) = x$ is unstable. For details see Section 5.

We work within first-order logic with equality. Let L be the language $\{+;\ min;\ C;\ \{|_n\}_n; \{\delta_c\}_c\}$, where $+$ and min are function symbols of arity 2, C is an infinitely countable set of constant symbols, $\{|_n\}_n$ and $\{\delta_c\}_c$ are infinitely countable sets of relation symbols of arity 1. We investigate the theory of the structure

$$\mathcal{A} = (\mathbb{Z}^\infty;\ +;\ min;\ C; \{|_n\}_{n\in\mathbb{N}};\ \{\delta_c\}_{c\in\mathbb{Z}^\infty}),$$

where

$$x + y = z \iff (x)_i + (y)_i = (z)_i, \text{ for all } i \in \mathbb{Z},$$
$$min(x,y) = z \iff min((x)_i,(y)_i) = (z)_i, \text{ for all } i \in \mathbb{Z},$$
$$C \text{ is a set of constants, exactly one for each element of } \mathbb{Z}^\infty,$$
$$|_n(x) \text{ if and only if } n \text{ divides } (x)_i, \text{ for all } i \in \mathbb{Z},$$
$$\delta_c(x) \iff \text{for all } i \in \mathbb{Z} \text{ if } (c)_i \neq 0 \text{ then } (c)_i \text{ divides } (x)_i.$$

In Section 2 we give an effective reduction of the problem of truth of existential formulae in $\mathcal{A}$ to that of solvability of systems of equations and inequations in Presburger Arithmetic. As a result we obtain the following

Theorem 1. *The existential theory of $\mathcal{A}$ is decidable.*

An (unknown to us) referee has pointed out the following: The structure $\mathcal{A}$ can be effectively interpreted in the structure $\mathcal{B} = (\mathbb{Q}^+, \times, N, <_P)$, where N is a 1-place predicate standing for natural numbers, and $<_P$ is a 2-place predicate standing for usual order relation in $\mathbb{Q}^+$ restricted on primes. F. Maurin in [6] proved that the first order theory of $(\mathbb{N}, \times, <_P)$ is decidable. One can show that the decidability of $Th(\mathcal{B})$ follows from [6]. We describe this interpretation in Section 3. So we deduce

Theorem 2. *The theory of $\mathcal{A}$ is decidable.*

Since Theorem 1 follows from Theorem 2, we say few words about complexity of algorithm given in proof of Theorem 1 and the complexity for the translation of the existential fragment of $\mathcal{A}$ into $\mathcal{B}$. In [11] has been proved the following fact on complexity of Presburger Arithmetic:

Theorem 3. *There exists a Q.E. procedure assigning to any prenex formula φ (in the language of Presburger Arithmetic) an equivalent quantifier free formula φ'. If φ has at most a quantifier-blocks each of length at most b, then the algorithm runs in time and space bounded by $2^{2^{c\cdot length(\varphi)^{(4b)^a}}}$ for some positive constant c.*

Combining the following facts: Theorem 3 with a=1 and that the reduction of the existential theory of $\mathcal{A}$ to existential theory of Presburger Arithmetic is in time $O(2^n)$, where n is the length of input (i.e., of the existential sentence), we obtain the time complexity for our algorithm to be $O(2^{2^{length(\varphi)4b}})$.

On the other hand, the translation of any existential L-sentence φ into a $\{\times, N, <_P\}$-sentence φ' gives us that φ' is of depth 3. Using the complexity bounds given in [6] for sentences of depth 3, we obtain time complexity of the algorithm deciding φ' be $O(2^{2^{2^{length(\varphi')}}})$.

Therefore the complexity of our algorithm for the existential fragment of $\mathcal{A}$ seems to be lower in comparison to that obtained by translating existential theory of $\mathcal{A}$ into the structure $\mathcal{B}$.

In Section 4 we consider the following extension of $\mathcal{A}$: let G be a finite abelian group and $a : \mathbb{Z}^\infty \to G$ a recursive epimorphism of groups. For each $g \in G$ we consider the predicate P_g, of arity 1 and interpret $P_g(x)$ as $a(x) = g$. We augment the structure $\mathcal{A}$ by the set $\{P_g :\ g \in G\}$ to $\mathcal{A}_a$. We show

Theorem 4. *The existential theory of $\mathcal{A}_a$ is decidable.*

In Section 5 we note that

Theorem 5. *$Th(\mathcal{A})$ satisfies the independence property (*IP*), and is therefore unstable.*

For more decidability results concerning Skolem Arithmetic and extensions of Skolem arithmetic see [10], [5], [3] and [1].

2 Existential theory of the structure $\mathcal{A}$

In this section we describe a simple algorithm which decides whether given existential sentence is true or not in $\mathcal{A}$ by solving a concrete (natural) number of systems in Presburger Arithmetic.

Remarks and notations.

- (i) Subtraction is definable, by $x-y=z \iff x=y+z$, therefore we use the symbol of subtraction whenever it is required.
- (ii) Some properties of $\mathbb{Z}$ are inherited by $\mathbb{Z}^\infty$. We list some of them, which are going to be used later.
  - (ii-a) Let $x,y,z,w \in \mathbb{Z}^\infty$, then $min(x,y)=z \iff min(x+w,y+w)=z+w$.
  - (ii-b) Let $x,y \in \mathbb{Z}^\infty$, with $min(x,y)=y$, then $(y)_i \leq (x)_i$, for all $i \in \mathbb{Z}$.
  - (ii-c) A consequence of (ii-b) is that non-negative sequence x is defined by $min(x,\mathbf{0})=\mathbf{0}$, where $\mathbf{0}$ is a constant sequence with all co-ordinals equal to 0.
- (iii) Every quantifier-free formula in L, with variables $\bar{x}=(x_1,...,x_n)$, is equivalent to a boolean combination of formulae of the form

$$\boxed{min(\pi_1(\bar{x}),\pi_2(\bar{x}))=\pi_3(\bar{x})},\ \boxed{\pi(\bar{x})=0},\ \boxed{\delta_c(\pi(\bar{x}))},\ \boxed{|_n(\pi(\bar{x}))}$$

  where π, π_1, π_2, and π_3 are polynomials over $\mathbb{Z}^\infty$ of degree one and c is a constant.
- (iv) We call a formula *sentence* if no variable that occurs in it is free.
- (v) Let $x \in \mathbb{Z}^\infty$. We denote by *Supp(x)* the support of the sequence x, i.e., $Supp(x)=\{i \in \mathbb{Z} : (x)_i \neq 0\}$.

Lemma 1. *Every existential sentence σ of L is equivalent to a finite disjunction of formulae of the form:*

$$\sigma_0 \wedge\ \exists \bar{x}\ [\sigma_1 \wedge \sigma_2 \wedge \sigma_3 \wedge \sigma_4 \wedge \sigma_5] \tag{3}$$

where σ_0 is a quantifier-free formula,

$$\sigma_1(\bar{x}): \bigwedge_i f_i(\bar{x})=\mathbf{0}\ , \tag{4}$$

$$\sigma_2(\bar{x}): \bigwedge_j f'_j(\bar{x})\neq\mathbf{0}\ , \tag{5}$$

$$\sigma_3(\bar{x}): \bigwedge_\rho min(\pi_\rho(\bar{x}),\pi'_\rho(\bar{x}))=\mathbf{0}\ , \tag{6}$$

$$\sigma_4(\bar{x}): \bigwedge_v \delta_{c_v}(h_v(\bar{x}))\ , \tag{7}$$

$$\sigma_5(\bar{x}): \bigwedge_\xi |_{n_\xi}(g_\xi(\bar{x})) \wedge \bigwedge_{\xi'} \not|_{n'_{\xi'}}(g'_{\xi'}(\bar{x}))\ , \tag{8}$$

where
each index among i, j, ρ, v, ξ, ξ' ranges over a finite set, each of $f_i, f'_j, \pi_\rho, \pi'_\rho, h_v, g_\xi, g'_{\xi'}$ is a degree-one polynomial of the indicated variables over $\mathbb{Z}$.

Proof. Given an existential sentence $\varphi = \exists \bar{x} \psi(\bar{x})$, the quantifier-free formula $\psi(\bar{x})$ can be written in disjunction-normal form with literals as mentioned in the remark (iii). The form of σ_3, i.e., where only $\mathbf{0}$ appears on the right-hand side of each relation min, is justified by the remark (ii-a). Note that some negations can be eliminated:

- $min(x, y) \neq z$ is equivalent to $\exists w(w = min(x, y) \wedge w - z \neq \mathbf{0})$. Observe that we define the negation of min by existential formula. That means that we increase the number of quantifiers by one, but this is not affecting the result.
- Let $c \in \mathbb{Z}^\infty$. Consider the following set:

$$\mathcal{C}_c = \{c' \in \mathbb{Z}^\infty : c' \neq \mathbf{0} \text{ and for all } i \in \mathbb{Z},\ 0 \leq (c')_i < |(c)_i|\},$$

  where by $|(c)_i|$we denote the usual absolute value in $\mathbb{Z}$. Therefore $\neg\delta_c(x)$ is equivalent to $\bigvee_{c' \in \mathcal{C}_c} \delta_c(x + c')$.

This do not work for $\not|_n(x)$, when x is a variable, because we do not know *a priori* the support of x.

Lemma 2. *With notation as in the conclusion of Lemma 1, the formula $\sigma_1(\bar{x})$ can be omitted.*

Proof. Without loss of generality, we assume that x_1 occurs (non-trivially) in first equation of σ_1. Then by multiplying by suitable integers (non-zero) we obtain x_1 to have the same coefficient in each of its occurrence in $\sigma_1 \wedge \cdots \wedge \sigma_5$. Let the first equation of σ_1 be $a_1x_1 + \cdots + a_nx_n + c = \mathbf{0}$, with $a_1 \in \mathbb{Z}$ and $a_1 \neq 0$. We substitute every occurrence of a_1x_1 in $\sigma_1 \wedge \cdots \wedge \sigma_5$ by $-a_2x_2 - \cdots - a_nx_n - c$. Then replace σ_5 with $\sigma_5 \wedge |_{a_1}(-a_2x_2 - \cdots - a_nx_n - c)$.

We continue by induction on the number of equalities.

Consider an open formula $\psi(\bar{x})$ of the form $\sigma_2 \wedge \cdots \wedge \sigma_5$, with notation given in Lemma 1. Let K be the set of all constants that occur in $\psi(\bar{x})$. Define $Supp(K) := \bigcup_{c \in K} Supp(c)$. Obviously K is a finite set. We denote the cardinality of K with $|K|$.

Let $a_1x_1 + \ldots a_nx_n + c$ be a polynomial with $a_i \in \mathbb{Z}$, $c \in \mathbb{Z}^\infty$ and variables $x_1, \ldots, x_n$. Then it is easy to check that for all $\kappa \in \mathbb{Z}$

$$(a_1x_1 + \cdots + a_nx_n + c)_\kappa = a_1(x_1)_\kappa + \cdots + a_n(x_n)_\kappa + (c)_\kappa. \tag{9}$$

Therefore it makes sense to define the following:

- For each n-tuple $\bar{x}$ of variables $x_1, \ldots, x_n$ and $\kappa \in \mathbb{Z}$ we define $(\bar{x})_\kappa = ((x_1)_\kappa, \ldots, (x_n)_\kappa)$.
- For each $\kappa \in \mathbb{Z}$ we define $\varphi_\kappa((\bar{x})_\kappa)$ to be the formula:

$$\bigwedge_v (c_v)_\kappa | (h_v(\bar{x}))_\kappa \wedge \bigwedge_\xi n_\xi | (g_\xi(\bar{x}))_\kappa \wedge$$

$$\wedge \bigwedge_\rho [[(\pi_\rho(\bar{x}))_\kappa = 0 \wedge (\pi'_\rho(\bar{x}))_\kappa \geq 0\ ] \vee [(\pi_\rho(\bar{x}))_\kappa \geq 0 \wedge (\pi'_\rho(\bar{x}))_\kappa = 0\ ]].$$

Lemma 3. *For all $\kappa \notin Supp(K)$ the formulae φ_κ are the same formula.*

Proof. For every $\kappa \notin Supp(K)$ we have that $(c)_\kappa = 0$, for every constant c that occurs in σ. Let $i : 1 \leq i \leq n$, then observe that due to (9) we have that the corresponding coefficients of $(x_i)_\kappa$ are the same for all $\kappa \in \mathbb{Z}$. This is enough to obtain that all φ_κ, for $\kappa \notin Supp(K)$, are equal.

Let Σ_5 be the set of $\not|_{n'_{\xi'}}$ which occur in σ_5. Let Σ_2 be the set of inequalities in σ_2. A partition of $\Sigma_2 \cup \Sigma_5$ of length λ is a set of λ non-empty subsets of $\Sigma_2 \cup \Sigma_5$, say $D_1, \dots, D_\lambda$ such that $\bigcup_{i=1}^{\lambda} D_i = \Sigma_2 \cup \Sigma_5$ and $D_i \cap D_j = \emptyset$ for every $i \neq j$. Consider D to be the set of all partitions of any length of the set $\Sigma_2 \cup \Sigma_5$. The set of all partitions of length λ of the set $\Sigma_2 \cup \Sigma_5$ we will denote with $D^{(\lambda)}$.

Fix $\lambda \in \{1, \dots, |\Sigma_2 \cup \Sigma_5|\}$. Let K' be any finite extension of K such that $|K'| - |K| \geq |\Sigma_2 \cup \Sigma_5| + 1$ and fix $\kappa_1, \dots, \kappa_\lambda \in K'$. We define the formula $\psi_{\kappa_1,\dots,\kappa_\lambda}((\bar{y})_{\kappa_1}, \dots, (\bar{y})_{\kappa_\lambda})$ in $n \cdot \lambda$ variables as follows:

$$\Big(\bigvee_{\{D_1,\dots,D_\lambda\} \in D^{(\lambda)}}\Big)\Big[\bigwedge_{i=1}^{\lambda}\Big(\bigwedge_{t \in D_i} t((\bar{y})_{\kappa_i})\Big)\Big].$$

Lemma 4. *Using the notation of Lemma 1 and of the discussion above, the following are equivalent:*
(a) $\exists \bar{x} = (x_1, \dots, x_n) \in (\mathbb{Z}^\infty)^n [\sigma_2(\bar{x}) \wedge \dots \wedge \sigma_5(\bar{x})]$,
(b) $(\exists (\bar{y})_j \in \mathbb{Z}^n)_{j \in K'} \Big[\Big(\bigwedge_{\kappa \in K'} \varphi_\kappa((\bar{y})_k\Big) \wedge \Big(\bigvee_{\lambda=1}^{|\Sigma_2 \cup \Sigma_5|} [\bigvee_{(\kappa_1,\dots,\kappa_\lambda)} \psi_{\kappa_1,\dots,\kappa_\lambda}((\bar{y})_{\kappa_1}, \dots, (\bar{y})_{\kappa_\lambda})]\Big)\Big]$.

Proof. $(a) \Rightarrow (b)$ Let $t_1, \dots, t_{|\Sigma_2 \cup \Sigma_5|}$ be a list of the elements of $\Sigma_2 \cup \Sigma_5$.
Assume that there are some $x_1, \dots, x_n \in \mathbb{Z}^\infty$ for which $\sigma_2 \wedge \dots \wedge \sigma_5$ holds. Let K' as we defined previously. Consider the following cases:

Case 1: Assume that $\bigcup_{i=1}^{n} Supp(x_i) \subseteq K'$. For every $t_i \in \Sigma_2 \cup \Sigma_5$ define $\mu_i = \min\{k \in K' : t_i((\bar{x})_k) \text{ holds}\}$. Let $\mu_1, \dots, \mu_{|\Sigma_2 \cup \Sigma_5|}$ be the produced sequence. Then we define by induction a partition of $\Sigma_2 \cup \Sigma_5$ as follows:

$$D_1 = \{t_j \in \Sigma_2 \cup \Sigma_5 : t_j((\bar{x})_{\mu_1}) \text{ holds}\},$$
$$D_{i+1} = \{t_j \in \Sigma_2 \cup \Sigma_5 : t_j((\bar{x})_{\mu_{i+1}}) \text{ holds}\} \setminus D_i.$$

Let λ be such that $D_\lambda \neq \emptyset$ and $D_{\lambda+1} = \emptyset$. It easy to see that $\{D_1, \dots, D_\lambda\}$ is a partition of $\Sigma_2 \cup \Sigma_5$, $\mu_i \in K'$ and that $\psi_{\mu_1,\dots,\mu_\lambda}((\bar{x})_{\mu_1}, \dots, (\bar{x})_{\mu_\lambda})$ holds. Also for every $\kappa \in \mathbb{Z}$ we have that $\varphi_\kappa((x_1)_\kappa, \dots, (x_n)_\kappa)$ holds true. Therefore $\varphi_\kappa((x_1)_\kappa, \dots, (x_n)_\kappa)$ holds for every $\kappa \in K'$.

Case 2: Assume that $\bigcup_{i=1}^{n} Supp(x_i) \not\subseteq K'$. Let $M = \bigcup_{i=1}^{n} Supp(x_i)$. For every $t_i \in \Sigma_2 \cup \Sigma_5$ define $\widetilde{\mu_i} = \min\{k \in K \cup M : t_i((\bar{x})_k) \text{ holds}\}$. Let $\widetilde{\mu_1}, \dots, \widetilde{\mu}_{|\Sigma_2 \cup \Sigma_5|}$ be the produced sequence. Then we define by induction a partition of $\Sigma_2 \cup \Sigma_5$ as follows:

$$D_1 = \{t_j \in \Sigma_2 \cup \Sigma_5 : t_j((\bar{x})_{\widetilde{\mu_1}}) \text{ holds}\},$$
$$D_{i+1} = \{t_j \in \Sigma_2 \cup \Sigma_5 : t_j((\bar{x})_{\widetilde{\mu}_{i+1}}) \text{ holds}\} \setminus D_i.$$

Let λ be such that $D_\lambda \neq \emptyset$ and $D_{\lambda+1} = \emptyset$. Similarly to the first case, $\{D_1, \dots, D_\lambda\}$ is a partition of $\Sigma_2 \cup \Sigma_5$. Note that $\lambda \leq |\Sigma_2 \cup \Sigma_5|$, i.e., there is a correspondence of $\{\widetilde{\mu_1}, \dots, \widetilde{\mu_\lambda}\}$ with some $\{\mu_1, \dots, \mu_\lambda\} \subset K'$ such that for all $i \in \{1, \dots, \lambda\}$ if $\widetilde{\mu_i} \in K$ then $\mu_i = \widetilde{\mu_i}$. Consider $y_1, \dots, y_n \in \mathbb{Z}^\infty$ defined as follows:

$$(y_i)_j = \begin{cases} (x_i)_{\widetilde{\mu}_\rho}, & \text{if } j = \mu_\rho \notin K, \text{ for some } \rho \in \{1, \dots, \lambda\} \\ (x_i)_j, & \text{else,} \end{cases} \tag{10}$$

We have that $\varphi_\kappa((\bar{x})_\kappa)$ holds true for every $\kappa \in \mathbb{Z}$. Therefore $\varphi_\kappa((\bar{y})_\kappa)$ holds for every $\kappa \in \mathbb{Z}$. Consequently, $\varphi_\kappa((y_1)_\kappa, \dots, (y_n)_\kappa)$ holds for every $\kappa \in K'$. For the partition $\{D_1, \dots, D_\lambda\}$ and $\mu_1, \dots, \mu_\lambda \in K'$ as given above, we have that $\psi_{\mu_1,\dots,\mu_\lambda}((\bar{y})_{\mu_1}, \dots, (\bar{y})_{\mu_\lambda})$ holds.

$(b) \Rightarrow (a)$ Assume that for $j \in K'$ there are $(\bar{y})_j \in \mathbb{Z}^n$ such that $\bigwedge_{\kappa \in K'} \varphi_\kappa((\bar{y})_k)$ and there is a partition $\{D_1, \dots, D_\lambda\}$ of $\Sigma_2 \cup \Sigma_5$ and some $\kappa_1, \dots, \kappa_\lambda \in K'$ such that $\psi_{\kappa_1,\dots,\kappa_\lambda}((\bar{y})_{\kappa_1}, \dots, (\bar{y})_{\kappa_\lambda})$ holds true.
For $i = 1, ..., n$ we define the following elements of $\mathbb{Z}^\infty$:

$$(x_i)_j = \begin{cases} (y_i)_j, & \text{if } j \in K', \\ 0, & \text{else,} \end{cases} \tag{11}$$

We show that for these $x_1, \dots, x_n$ the formula $\sigma_2 \wedge \dots \wedge \sigma_5$ holds true.

($\mathbf{F_1}$) For every $\kappa \notin K'$ we have that $(c)_\kappa = 0$, for every constant c that occurs in σ. The same holds for each $(x_i)_\kappa$ by definition of x_i, i.e., $\pi_\rho, \pi'_\rho, h_v, g_\xi$ (as given in Lemma 1) are all equal to 0. Therefore $[(\sigma_3(\bar{x}))_\kappa] \wedge [(\sigma_4(\bar{x}))_\kappa] \wedge [\bigwedge_\xi \ n_\xi|(g_\xi(\bar{x}))_\kappa]$, for every $\kappa \notin K'$. Combining this latter fact with hypothesis for $\varphi_\kappa((y_1)_\kappa, \ldots, (y_n)_\kappa)$ for every $\kappa \in K'$ together with definition of x_i for $\kappa \in K'$ we obtain that

$$(\sigma_3(\bar{x}))_\kappa \wedge (\sigma_4(\bar{x}))_\kappa \wedge \bigwedge_\xi \ n_\xi|(g_\xi(\bar{x}))_\kappa, \text{ for every } \kappa \in \mathbb{Z}.$$

($\mathbf{F_2}$) Let $t_i \in \Sigma_2 \cup \Sigma_5$, with $i \in \{1, \ldots, |\Sigma_2 \cup \Sigma_5|\}$. Then there is $j \in \{1, \ldots, \lambda\}$ such that $t_i \in D_j$. Then by hypothesis we have that $\bigwedge_{\tau \in D_j}(\tau(\bar{y}))_{\kappa_j}$. Thus $(t_i(\bar{y}))_{\kappa_j}$. Observe that t_i is a atomic formula containing (exactly) one negation, i.e.,

$$\forall \bar{y} \in (\mathbb{Z}^\infty)^n [\ t_i(\bar{y}) \iff (t_i(\bar{y}))_\kappa \text{ for some } \kappa \in \mathbb{Z} \ .]$$

Combining the latter together with the definition of x_i we obtain that

$$(\sigma_2(\bar{x})) \wedge \bigwedge_{\xi'} \ \not|_{n'_{\xi'}}(g'_{\xi'}(\bar{x})) \ ,$$

By ($\mathbf{F_1}$), ($\mathbf{F_2}$) we obtain the required, i.e., $\sigma_2(\bar{x}) \wedge \cdots \wedge \sigma_5(\bar{x})$

Theorem 6. *The existential theory of $\mathcal{A}$ is decidable.*

Proof. Observe that due to Lemmas 1, 2 and 4, it is enough to give a decision-algorithm for any existential sentence in L. The complexity of the reduction is exponential because of the use of partition. Also exponential time is needed for solving the corresponding problem over $\mathbb{Z}$.

3 The theory of $\mathcal{A}$

F. Maurin proved in [6] that the first order theory of $(\mathbb{N}, \times, <_P)$, where $<_P$ is a 2-place predicate standing for the usual order relation in $\mathbb{N}$ restricted on primes, is decidable. Consider the structure $\mathcal{B} = (\mathbb{Q}^+, \times, N, <_P)$, where N is a 1-place predicate standing for the set of natural numbers. The decidability of Th($\mathcal{B}$) follows from the decidability of $(\mathbb{N}, \times, <_P)$.

We will interpret the structure $\mathcal{A}$ into $\mathcal{B}$. To do this we interpret each element of $\mathbb{Z}^\infty$ as a rational number, as in the Introduction (for details see below). Towards this task we define in $\mathcal{B}$ the constant 1, the predicate $Prime(x)$ (which stands for "x is a prime"), $S_P(x, y)$ (which stands for "y is the next prime, greater than the prime x") and the predicate $P_i(x)$ (which stands for "x is the i-th prime").

- $x = 1 \iff N(x) \wedge \forall y(x \times y = y \times x = y)$.
- $Prime(x) \iff N(x) \wedge \forall y, z(x = y \times z \to (y = 1 \vee z = 1))$.
- $S_P(x, y) \iff Prime(x) \wedge Prime(y) \wedge \forall z(Prime(z) \to \neg(x <_P z \wedge z <_P y))$.
- $P_0(x) \iff Prime(x) \wedge \forall y(Prime(y) \to (x <_p y \vee x = y))$.
- $P_{i+1}(x) \iff \exists y_1, ..., y_i(\bigwedge_j Prime(y_j) \wedge Prime_0(y_0) \wedge y_i = x \wedge \bigwedge_j S_P(y_j, y_{j+1}))$.

As we mentioned in the Introduction, the function 'gcd'(x, y) is not definable from $|$ (divisibility) when $x, y \in \mathbb{Q}^+$, but for $x, y \in \mathbb{N}$

- $d = gcd(x, y) \iff$

$N(x) \wedge N(y) \wedge \exists z_1(d \times z_1 = x) \wedge \exists z_2(d \times z_2 = y) \wedge \forall w[\exists z_3(d \times z_3 = x) \wedge \exists z_3(d \times z_3 = y \to \exists z_4(w \times z_4 = d))]$.

Similarly we define $lcm(x, y)$ for $x, y \in \mathbb{N}$, i.e.,

- $e = lcm(x, y) \iff$

$N(x) \wedge N(y) \wedge \exists z_1(x \times z_1 = e) \wedge \exists z_2(y \times z_2 = e) \wedge \forall w[\exists z_3(x \times z_3 = e) \wedge \exists z_3(y \times z_3 = e \to \exists z_4(e \times z_4 = w))]$.

Finally for every $x \in \mathbb{Q}^+$, with $x = \frac{y}{z}$ and $(y, z) = 1$ we define the denominator z of x, $den(x)$:

- $den(x) = z \iff N(z) \wedge N(x \times z) \wedge \forall w_1[N(w) \wedge N(w_1 \times x) \to \exists w_2(w_2 \times z = w_1)]$.

The structure $\mathcal{A}$ is definable in $\mathcal{B}$ as follows:

- $d =$'gcd'(x, y) is interpreted by $\frac{gcd(lcm(den(x),den(y))\times x,\quad lcm(den(x),den(y))\times y)}{lcm(den(x),den(y))}$.
- For each constant c of $\mathcal{A}$ with support $\{i_1, ..., i_n\}$ we have $x = c$ is interpreted by $\exists y_1, ..., y_n(\bigwedge_j P_j(y_j) \wedge x = y_1^{(c)_{i_1}} \times ... \times y_n^{(c)_{i_n}})$.
- $|_n(x)$ is interpreted by $\exists y(y^n = x)$.

According to [6] we obtain:

Theorem 7. *The full theory of $\mathcal{A}$ is decidable.*

4 The existential theory of the structure $\mathcal{A}_a$

Let G be a finite abelian group, and $a : \mathbb{Z}^\infty \to G$ a recursive epimorphism of groups. Consider L_a to be the language $\{+;\ min;\ C; \{|_n\}_{n\in\mathbb{N}}; \{P_g\}_{g\in G},\ \ \{\delta_c\}_{c\in\mathbb{Z}^\infty}\}$, where $+$, min, C, δ_c and $|_n$ are as given in section 1. Each P_g is a unary predicate. We are interested in the existential theory of the structure

$$\mathcal{A}_a = (\mathbb{Z}^\infty;\ +;\ min;\ C; \{|_n\}_{n\in\mathbb{N}}; \{P_g\}_{g\in G},\ \{\delta_c\}_{c\in\mathbb{Z}^\infty}),$$

where $+$, min, C, δ_c and $|_n$ are interpreted as in section 1 and

$$P_g(x) \iff a(x) = g.$$

Since $|G : Ker(a)| = |Im(a)|$, we have that the kernel of a has a finite index in G.

Lemma 5. *Let H be a subgroup of $\mathbb{Z}^\infty$ of finite index. Then there are $i_1, ..., i_k \in \mathbb{Z}$ and $n_{i_1}, ..., n_{i_k} \in \mathbb{N}$ such that:*

$$H = \{x \in \mathbb{Z} : (x)_{i_j} \in n_{i_j}\mathbb{Z}\}.$$

Proof. Since $\mathbb{Z}^\infty$ is an abelian group, we can define the corresponding quotient group $\mathbb{Z}^\infty/H$ which is finite (and abelian) of order $[\mathbb{Z}^\infty : H] = m$. On the other hand, H as a subgroup of direct sum is of a form $\bigoplus_{i\in\mathbb{Z}} n_i\mathbb{Z}$ (since any subgroup of $\mathbb{Z}$ is of the form $n\mathbb{Z}$ for some $n \in \mathbb{Z}$). Let I_H be the set of all i such that $n_i \neq 1$ in the direct sum of H. Consider $a_i \in \mathbb{Z}^\infty$ such that $(a_i)_i = 1$ and $(a_i)_j = 0$ for all integers $j \neq i$. Then $a_i + H$ are distinct cosets for all $i \in I_H$. Since H is of finite index in $\mathbb{Z}^\infty$, we have only finitely many distinct cosets of H in $\mathbb{Z}^\infty$. Thus I_H is a bounded set, i.e., $I_H = \{i_1, ..., i_k\}$. Therefore, for the corresponding $n_{i_1}, ..., n_{i_k} \in \mathbb{N}$ we have

- $\prod_j n_{i_j} = m$,
- $\mathbb{Z}^\infty/H \cong \mathbb{Z}_{n_{i_1}} \oplus \cdots \oplus \mathbb{Z}_{n_{i_k}}$ and
- $H = \{x \in \mathbb{Z} : (x)_{i_j} \in n_{i_j}\mathbb{Z}\}$.

Theorem 8. *The existential theory of $\mathcal{A}_a$ is decidable.*

Proof. First observe that it is enough to have in L_a only one predicate of the form P_g, and concretely P_e, where e is the identity element of G. Next we show that the predicate P_e is definable in L by an open formula. Indeed, by repeating the arguments in Section 2 we can determine I_H as defined in Lemma 5. Next consider the constant c_H which corresponds to the element in $\mathbb{Z}^\infty$ satisfying $(c^{\mathcal{A}_a})_{i_j} = n_{i_j}$ for every $i_j \in I_H$ and 0 everywhere else. Then

$$P_e(x) \iff |_{c_H} x.$$

Thus we deduce the decidability of the existential theory of $\mathcal{A}_a$ from Theorem 7.

Using Lemma 5 and similar arguments as in Theorem 8 we can extend the decidability result in the following sense: let F be the set of all subgroups H of $\mathbb{Z}^\infty$ of finite index in $\mathbb{Z}^\infty$ and let $F' \subseteq F$. Consider the structure $\mathcal{A}_{F'} = (\mathbb{Z}^\infty;\ +;\ min;\ C; \{|_n\}_{n\in\mathbb{N}}; \{P_H\}_{H\in F'})$, where $+$, min, C and $|_n$ are interpreted as usual and $P_H(x) \iff x \in H$.

Corollary 1. *The existential theory of $\mathcal{A}_{F'}$ is decidable.*

5 Properties of $\mathcal{A}$

We would like to say few words on some properties of $\mathcal{A}$ from the model theoretical point of view. We start with the following theorem of Shelah (see [2]).

Theorem 9. *Let T be a complete theory. Then T is unstable if and only if there is a model $\mathcal{M}$ of T, with universe M, an infinite $X \subset M^n$ and a formula $\varphi(\bar{x}, \bar{y})$ ($\bar{x} = (x_1, ..., x_n)$, $\bar{y} = (y_1, ..., y_n)$) defining total ordering on X.*

By its nature, the function *min* induces an ordering. Therefore we obtain

Proposition 1. *Let T be a theory of any structure $\mathcal{S}$ of the form $(\mathbb{Z}^\infty, min, S_1, S_2, S_3)$, where S_1 is a set of relations, S_2 a set of functions and S_3 a set of constants (some S_i might be an empty set). Then T is unstable.*

Proof. Consider $\varphi(x, y)$ to be $min(x, y) = x$. Let $\mathcal{M}$ be the given structure $(\mathbb{Z}^\infty, min, S_1, S_2, S_3)$ and define $X = \{a_i : i \in \mathbb{N}\}$, where $a_i \in \mathbb{Z}^\infty$ such that $(a_i)_0 = i$ and for all $j \neq 0$ $(a_i)_j = 0$. Then it is easy to check that X is infinite subset of $\mathbb{Z}^\infty$ and that φ defines a total ordering on X.

Corollary 2. *$Th(\mathcal{A})$ is unstable.*

Another notion that is worth to consider is the *independence property*. Namely,

Definition 1. *Let T be a complete theory. We say that a formula $\varphi(\bar{x}, y)$ ($\bar{x} = (x_1, ..., x_m)$) satisfies* IP *(independence property) in T if and only if in every model M of T there is for each $n \in \mathbb{N}$ a family $b_0, ...b_{n-1}$ such that, for all subsets X of $\{0, ..., n-1\}$ there is $(\bar{a}) \in |M|^m$*

$$M \models \varphi(\bar{a}, b_i) \iff i \in X.$$

T is said to satisfy IP *if there is a formula which satisfies* IP *in T.*

Proposition 2. *Let T be a complete theory as given in the statement of Proposition 1. Then T satisfies* IP*.*

Proof. We start by giving an equivalent definition of IP which can be found in [9]. Fix a formula $\varphi(\bar{x}, \bar{y})$ and denote the power set of $n = \{0, ..., n-1\}$ by $\mathcal{P}(n)$. Let I_n be the axiom

$$(\exists \bar{x}_i)_{i \in n}, (\exists \bar{y}_W)_{W \in \mathcal{P}(n)} [\bigwedge_{i \in W} \varphi(\bar{x}_i, \bar{y}_W) \wedge \bigwedge_{i \notin W} \neg\varphi(\bar{x}_i, \bar{y}_W)].$$

In order to prove that T satisfies IP, it is enough to prove for some $\varphi(\bar{x}, \bar{y})$ that $T \cup \{\mathrm{I}_n : n \in \mathbb{N}\}$ is consistent. Consider $\varphi(\bar{x}, \bar{y})$ be a formula just in two variables, i.e., $\bar{x} = x$, $\bar{y} = y$, with

$$\varphi(x, y) : \quad min(x, y) = x.$$

We claim that the structure $\mathcal{S}$, as given in the statement of Proposition 1, is a model of $T \cup \{\mathrm{I}_n : n \in \mathbb{N}\}$. Indeed, fix $n \in \mathbb{N}$ and for every $i \in n$ and $W \in \mathcal{P}(n)$, define

- $a_i \in \mathbb{Z}^\infty$ such that $(a_i)_i = 1$ and $(a_i)_j = 0$ for all integers $j \neq i$,
- $b_W \in \mathbb{Z}^\infty$ such that $b_W = \sum_{i \in W} a_i$.

Observe that if $i \in W$ then $(min(a_i, b_W))_i = 1 = (a_i)_i$ and for all $j \neq i$ $(min(a_i, b_W))_j = 0 = (a_i)_j$. While if $i \notin W$ then $(min(a_i, b_W))_i = 0 \neq (a_i)_i$. Thus

$$\mathcal{S} \models \bigwedge_{i \in W} \varphi(a_i, b_W) \wedge \bigwedge_{i \notin W} \neg\varphi(a_i, b_W)$$

Corollary 3. *$Th(\mathcal{A})$ satisfies* IP*.*

Thus another proof of Corollary 2 can be obtained by combining Corollary 3 and the following

Proposition 3. *([8]) Let T be a complete theory. Then if T satisfies* IP*, T is unstable.*

Lemma 6. *Let $\mathcal{B}$ any structure with universe $\mathbb{Z}^\infty$ and let the language of $\mathcal{B}$ contain the function (or the graph of the function) min with interpretation given in section 1. Then the theory of $\mathcal{B}$ satisfies* IP*, therefore it is unstable.*

6 Acknowledgment

The author would like to thank Th. Pheidas, D. Macpherson and A. Pillay for the enlightening discussions concerning this work and the referee for pointing out the definability of $\mathcal{A}$ in $\mathcal{B}$.

References

1. A. Bès, *A Survey of Arithmetical Definability*, a tribute to Maurice Boffa, pages 1-54. Soc. Math. Belgique, (2002).
2. C.C. Chang, H.J. Keisler *Model Theory*, Studies in Logic and the Foundations of Mathematics, (1990).
3. S. Feferman and R.L. Vaught, *The first order properties of products of algebraic systems*, Fundamenta Mathematicae, 47, (1959), 57-103.
4. E.R. Fisher, *Powers of saturated modules*, J. Symbolic Logic, 37(4) (1972), 777.
5. J.A. Makowsky, *Algorithmic uses of the FefermanVaught Theorem* , Annals of Pure and Applied Logic, Volume 126, Issues 1-3, (2004), 159 -213.
6. F. Maurin, *The theory of integer multiplication with order restricted to primes is decidable*, J. Symbolic Logic 62 (1), (1997), 123-130 .
7. A. Mostowski, *On Direct Products of Theories*, J. Symbolic Logic Volume 17 (3), (1952), 203-204.
8. A. Pillay, *An Introduction to Stability Theory*, Dover Publications, (2008).
9. Poizat, *A Course in Model Theory: An Introduction to Contemporary Mathematical Logic*, Springer, (2000).
10. T. Skolem, Uber gewisse Satzfunktionen in der Arithmetik, Skr. Norske Videnskaps-Akademie i Oslo, 7 , (1930).
11. V. Weispfenning *The complexity of almost linear diophantine problems*, J. Symbolic Computation, 10 (5), (1990), 395-403.

Formal Representations of Pronouns Using Universal Networking Language

Velislava Stoykova

Institute for Bulgarian Language, Bulgarian Academy of Sciences,
52, Shipchensky proh. str., bl. 17, 1113 Sofia, Bulgaria
email: vstoykova@yahoo.com

Abstract. The paper analyzes the use of semantic networks for representation of pronominal information for machine translation purposes. It discuss semantics, grammar features and some general principles and related problems for formal representations of pronouns in semantic networks. The formal representations of pronouns for two languages in the frameworks of Universal Networking Language (UNL) are analyzed with respect to linguistic motivation used for encodings. Finally, more general conclusions about formal representations of pronominal information for multilingual applications are drawn.
Keywords: Natural Language Processing, Computational Linguistics, Knowledge Representation, Semantic Networks, Machine Translation.

1 Introduction

Pronouns exist in all European languages and they share similar semantics and grammar features. As a knowledge representation task, they have been represented using mostly syntactic formal theories. At the same time, the semantic networks were successfully used to represent both lexical and grammar information with multilingual application Some of them like DATR language for lexical knowledge representation [4], WordNet [5] and Universal Networking Language (UNL) [11] have multilingual applications.

As for example DATR, there are programming applications made for lots of languages [3, 7, 8] which suggest various approaches and techniques that can be successfully used for multilingual application in machine translation. Further, we are going to analyze semantic and grammar features of pronouns and to use related approaches for UNL formal multilingual representations.

2 The semantics and grammar features of pronouns

Pronouns are usually analyzed with respect to their semantics, grammar features, functions and usage. The semantics of pronouns is connected to their roles to substitute, determine, relate and agree both with other words in the sentence or within the whole text. Formally, pronouns are usually represented with syntactic theories (as for English language). However, for some other languages the roles are realized by using subsequent grammar features of person, number, and gender (for some languages also case, definiteness, etc.). The related grammar features can be used for formal representations of pronouns by offering morpho-syntactic formal interpretation.

Moreover, the related grammar features are universal for all types of pronouns and are used for successful multilingual machine translation applications. In suggested approach they are used to present both grammar and lexical information by means of special linking hierarchical mechanism.

3 The Universal Networking Language

In the UNL approach, information conveyed by natural language is represented as a hypergraph composed of a set of directed binary labeled links (referred to as "relations") between nodes or hypernodes (the "Universal Words"(UWs)), which stand for concepts. UWs can also be annotated with "attributes" representing context information [11].

Universal Words (UWs) represent universal concepts and correspond to the nodes to be interlinked by "relations" or modified by "attributes" in a UNL graph. They can be associated to natural language open lexical categories (noun, verb, adjective and adverb). Additionally, UWs are organized in a hierarchy (the UNL Ontology), are defined in the UNL Knowledge Base and exemplified in the UNL Example Base, which are the lexical databases for UNL. As language-independent semantic units, UWs are equivalent to the sets of synonyms of a given language, approaching the concept of "synset" used by the WordNet.

Attributes are arcs linking a node onto itself. In opposition to relations, they correspond to one-place predicates, i.e., function that take a single argument. In UNL, attributes have been normally used to represent information conveyed by natural language grammatical categories (such as tense, mood, aspect, number, etc). Attributes are annotations made to nodes or hypernodes of a UNL hypergraph. They denote the circumstances under which these nodes (or hypernodes) are used. Attributes may convey three different kinds of information: (i) The information on the role of the node in the UNL graph, (ii) The information conveyed by bound morphemes and closed classes, such as affixes (gender, number, tense, aspect, mood, voice, etc), determiners (articles and demonstratives), etc., (iii) The information on the (external) context of the utterance. Attributes represent information that cannot be conveyed by UWs and relations.

Relations, are labeled arcs connecting a node to another node in a UNL graph. They correspond to two-place semantic predicates holding between two UWs. In UNL, relations have been normally used to represent semantic cases or thematic roles (such as agent, object, instrument, etc.) between UWs.

UNL-NL Grammars are sets of rules for translating UNL expressions into natural language (NL) sentences and vice-versa. They are normally unidirectional, i.e., the enconversion grammar (NL-to-UNL) or deconversion grammar (UNL-to-NL), even though they share the same basic syntax.

In the UNL Grammar there are two basic types of rules: (i) Transformation rules - used to generate natural language sentences out of UNL graphs and vice-versa and (ii) Disambiguation rules - used to improve the performance of transformation rules by constraining their applicability.

The UNL offers universal language-independent and open-source platform for multilingual applications [2]. The UNL application for English language is available but some applications for other languages like Russian [1] and Bulgarian [9, 6] are available as well.

3.1 Representing pronouns in UNL

The UNL offers various approaches to lexical knowledge representation including formal grammar rules, presentation of pronouns grammar features of person, number and gender, development of inflectional features (like for case and definiteness for some languages), etc.

Thus, UNL lexical knowledge representation scheme allows two types of transformation inflectional rules: (i) A-rules (affixation rules) apply over isolated word forms (as to generate possible inflections) and (ii) L-rules (linear rules) apply over lists of word forms (as to provide transformations in the surface structure). Affixation rules are used for adding morphemes to a given base form. They are used for generating inflections or derivations. There are two types of A-rules: (i) simple A-rules involve a single action (such as prefixation, suffixation, infixation and replacement), and (ii) complex A-rules involve more than one action (such as circumfixation).

There are four types of simple A-rules: (i) prefixation, for adding morphemes at the beginning of a base form, (ii) suffixation, for adding morphemes at the end of a base form, (iii) infixation, for adding morphemes to the middle of the base form, (iv) replacement, for changing the base form.

Further, we are going to analyze pronuns representation in the lexical database or UNL dictionary for English and Bulgarian by comparing examples mostly of personal, possessive, and reflexive pronouns.

The English pronouns formal representation uses mostly rules for syntactic transformations. Thus, the UNL dictionary uses grammar feature of person to link different types of pronouns by means of synonymic hierarchical lexical relations. It relates personal pronoun (for subject) I to personal pronoun (for object) me and to possessive adjective my and to possessive pronoun $mine$ and reflexive pronoun $myself$.

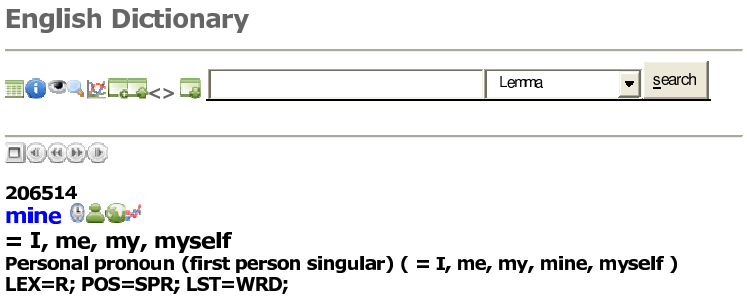


Fig. 1. The English possessive pronoun $mine$ in UNL representation.

Fig. 1 shows UNL dictionary entry for possessive pronoun $mine$ which uses hierarchical synonymic definition by relating grammar information like type of pronoun and grammar feature of person. It presents syntactic information through LEX=R and POS=SPR which are used by transformation grammar rules.

Fig. 2 presents UNL dictionary entry for reflexive pronoun $myself$ which includes same types of information structured the same way.

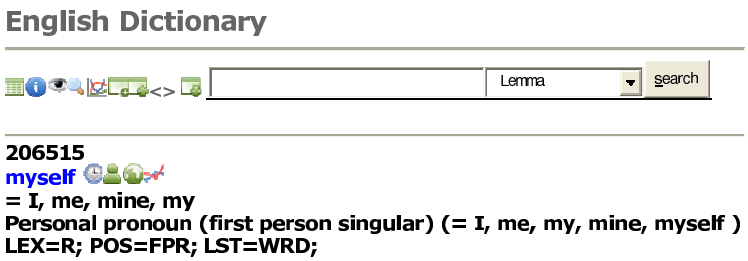


Fig. 2. The English reflexive pronoun $myself$ in UNL representation.

At the same time, there are languages like Bulgarian which present syntactic information for definiteness by using inflection. For that, pronouns are presented additionally by applying inflectional rules.

The formal representation of Bulgarian pronouns is a part of inflectional morphology application [9, 10] aimed to develop UNL grammar and lexical resources for several European languages. Thus, it uses some universal principles and offers interpretation of inflectional morphology which uses A-rules. Fig. 3 shows UNL dictionary entry for Bulgarian possessive pronoun moj.

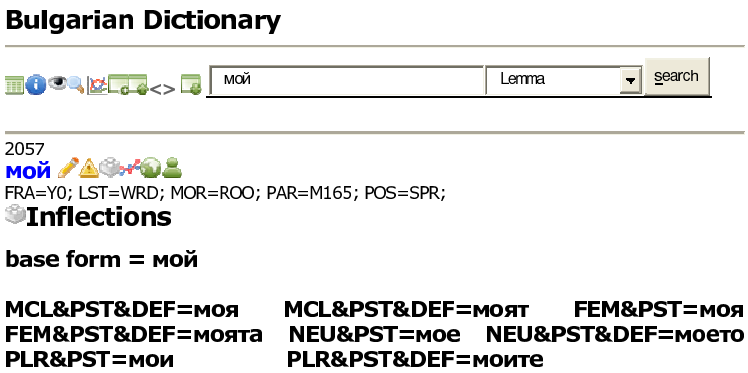


Fig. 3. The Bulgarian possessive pronoun moj in UNL representation.

It presents both lexical and grammar information. The grammar information is given by both syntactic and inflectional rules. The inflectional rules allow generation of all pronominal inflected forms and offer a sound alternations account mostly by the use

of A-rules. They are defined without the use of hierarchical inflectional representation. The transformation syntactic rules use specification POS=SPR to relate Bulgarian and English lexical entries for the same pronoun.

The sound alternations and the irregularity are interpreted within the definition of the main inflectional rule. The information about inflection is given through the specifier PAR=M165 which assign related inflectional type consisting of inflectional grammar rules (given at the Appendix).

The UNL application, also, represents a web-based intelligent information and knowledge management system which allows different types of semantic search with respect to various search criteria.

4 Conclusion

The formal UNL representations of English and Bulgarian pronouns use common formal framework based on interpretation of both lexical (synonymic hierarchical representation) and grammar features (like person, gender or definiteness) to relate pronouns for both languages. Additionally, they relate syntactic information by using common specifiers (like POS=SPR) to relate transformation syntactic rules through linking. The application is open for further improvement and development by introducing additional grammar rules and enlarging database.

References

1. Boguslavsky, I.: Some lexical issues of unl. In: J. Cardeosa, A. Gelbukh, E. Tovar (Eds.) Universal Network Language: Advances in Theory and Applications. Research on Computing Science. vol. 12, pp. 101–108 (2005)
2. Boitet, C., Boguslavskij, I., Cardenosa, J.: An evaluation of unl usability for high quality multilingualization and projections for a future unl++ language. In: In A. Gelbukh ed. Proceedings of CICLing, Lecture Notes in Computer Sciences. vol. 4394, pp. 361–373. Springer-Verlag (2007)
3. Corbett, G., Fraser, N.: Network morphology: a datr account of russian nominal inflection. Journal of Linguistics 29, 113–142 (1993)
4. Evans, R., Gazdar, G.: Datr: A language for lexical knowledge representation. Computational Linguistics 22(2), 167–216 (1996)
5. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press (1998)
6. Noncheva, V., Stancheva, Y., Stoykova, V.: The little prince project - encoding of bulgarian grammar. Tech. rep., www.undl.org, UNDL Foundation (2011)
7. Stoykova, V.: The definite article of bulgarian adjectives and numerals in datr. In: C. Bussler and D. Fensel, eds. Artificial Intelligence: Methodology, Systems, and Applications. LNAI 3192. pp. 256–266. Springer-Verlag (2004)
8. Stoykova, V.: Representing nominal inflectional morphology for slavonic languages in datr. In: H. Ganchev, B. Lowe, D. Norman, I. Soskov and M. Soskova, eds. Computability in Europe 2011. Models of Computation in Context. Abstract and Handout Booklet. pp. 201–209. St. Kliment Ohridsky University Press (2011)
9. Stoykova, V.: Bulgarian inflectional morphology in universal networking language. In: M. Kay and C. Boitet, eds. Proceedings of 24th International Conference on Computational Linguistics (COLING 2012): Demonstration Papers. pp. 423–430. ACL Anthology (2012)

10. Stoykova, V.: The inflectional morphology of bulgarian possessive and reflexive- possessive pronouns in universal networking language. In: Procedia Technology. vol. 1, pp. 400–406. Elsevier (2012)
11. Uchida, H., Zhu, M., Senta, T.D.: Universal Networking Language. UNDL Foundation (2005)

APPENDIX

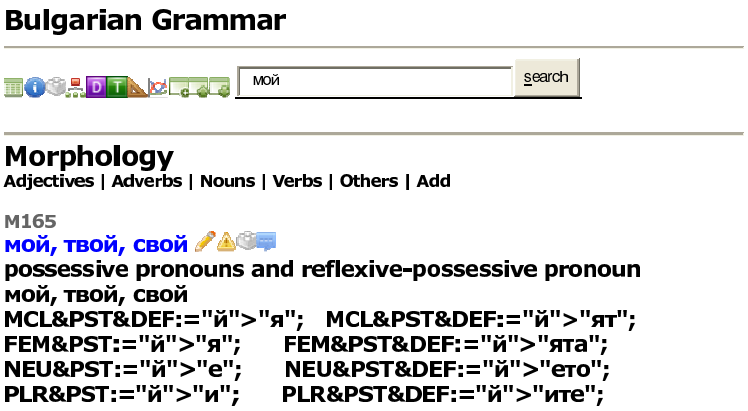


Fig. 4. The inflectional rules definitions for Bulgarian possessive pronoun moj.

The d-distance anticoloring Problem

S. Zucker

Department of Computer Science, Sapir Academic College, Israel

Abstract. Given an edge weighted graph G and positive integers x,y and d, the d-distance anticoloring problem asks about the existence of a partial vertex-coloring of G, with x vertices colored black and y white, such that each path between a black and a white vertex has cost at least d. We suggest a polynomial algorithm for solving this problem on weighted trees. We also present a sketch of an algorithm for solving this problem on chordal graphs.

1 Introduction

The *d-distance anticoloring problem* is a new problem, defined as follows. Given an undirected graph G with a cost function $C : E(G) \to \{1, 2, 3, \ldots\}$, positive integers x, y and $d > 1$, determine whether there exists a partial vertex-coloring of G such that x vertices are colored black and y vertices in white (with all other vertices left uncolored), such that there is no path P, with $|P| < d$, between any black and white vertex. Recall that $|P| = \sum_{e \in P} C(e)$.

This problem is in fact a generalization of the black-and-white coloring problem [10], in which $d = 2$. The NP-completeness of the d-distance anticoloring problem is derived from that of the black-and-white coloring problem, proved by Hansen *et al.* [8].

Prison safety can be considered as a motivation for this problem. A given prison detains prisoners from two different mafias. It is essentially important to keep the different prisoners in distanced cells from each other. When prisoners inhabit cells too close to each other riots break out. Obviously, prison cells correspond to nodes of the graph, and the two mafias to the two colors.

Hansen *et al.* [8] presented a polynomial algorithm for the original anticoloring problem in the case where G is a tree. In [10], there is an algorithm for that problem in the case where G is a chordal graph.

In this paper, we give an $O(n^3)$-algorithm for the d-distance anticoloring problem on trees. We also present a sketch of an algorithm for the case where G is chordal.

Section 2 presents formally the problem and states the main theorems. Section 3 gives a reduction which helps us solve the main problem. The rest of that section presents the algorithm of the new problem. Section 4 describes briefly a solution for the problem on chordal graphs.

2 Main Results

Let T be a tree with n vertices and let d be an integer. The attributes of a d-distance anticoloring of T are given by a pair (x, y), in which x is the number of black vertices and y the number of white vertices, with x and y satisfying the requirements of the problem. Thus, by having an array containing the maximal y for each value of x, we identify the attributes of all possible d-distance anticolorings. Such a pair is *non-dominated* if there exists no feasible d-distance anticoloring with x black and $y' > y$ white vertices. Our algorithm solves simultaneously the d-distance anticoloring problem for all pairs (x, y).

Problem 1.

Input: A rooted n-vertex tree T with a cost function $C : E(T) \to \{1, 2, 3, \ldots\}$ and an integer $d > 1$.

Output: An array dSOL which, for each $0 \leq x \leq n$, gives the maximal y such that there exists a $d-$distance anticoloring of T, with x black and y white vertices.

Theorem 1. *Problem 1 is solved in time $O(n^3)$.*

Note that this theorem gives a solution for all possible d-distance anticolorings. In fact, in order to find if there exists a d-distance anticoloring for a given (x, y), simply check if the algorithm returns a pair (x', y') with $x' \geq x$ and $y' \geq y$.

Problem 2.

Input: A rooted n-vertex chordal graph G and an integer $d > 1$.

Output: An array dSOL which, for each $0 \leq x \leq n$, gives the maximal y such that there exists a $d-$distance anticoloring of G, with x black and y white vertices.

Theorem 2. *Problem 2 can be solved in polynomial time.*

3 Proof of Theorem 1

3.1 A reduction of the Problem

We assume that the input tree T is rooted. For each vertex v, denote by T_v the subtree rooted at v.

To each vertex v of T we attach two arrays $v.B$ and $v.W$ and a three dimensional array $v.U$ – corresponding to feasible d-distance anticolorings of T_v in which v is black, white or uncolored, respectively. Thus, $v.B$ ($v.W$, respectively) contains for each value x the maximal possible y, assuming that v is colored black (white, respectively). Considering the array $v.U$, we need to record some more data. We define $v.U[1]$ to contain for each value x the maximal possible y, assuming that v is uncolored. The minimal distance of v from a black

(white, respectively) vertex $\widehat{v} \in T_v$ is save in $v.U[2]$ ($v.U[3]$, respectively). Note that there is no need to record such data in $v.B$ and $v.W$. For convenience, we assume in the following that these three arrays are saved in a common variable, named v.BWU.

We now show how an instance of Problem 1 can be reduced to an instance of the following Problem 3. For a given tree $T = (V, E)$, we create a new tree $T' = (V', E')$, with $V' = V \cup V_e$ and $E' = E \cup E_e$, where $V_e = \{v'_1, v'_2, \ldots, v'_{c-1} | e = (u, v) \in E(T), C(e) = c > 1\}$, and $E_e = \{(u', v'_1), (v'_1, v'_2), \ldots, (v'_{c-1}, v') | e = (u, v) \in E(T), C(e) = c > 1\}$. Meaning, for each edge $(u, v) \in E(T)$ with $C(u, v) = c > 1$, we add in T' new vertices $v'_1, v'_2, \ldots, v'_{c-1}$ and connect them with c unweighted edges $(u', v'_1), (v'_1, v'_2), \ldots, (v'_{c-1}, v')$. Obviously, T' is a tree and for each $(u, v) \in E(T)$ and $(u', v') \in E(T')$ we have that $C(u, v)$ is equal to the cost of the path between u' and v'. We define each of the new vertices to be left uncolored in T', i.e., if Φ is the coloring function, then $\Phi(v_i) = \texttt{uncolored}$ for each $1 \leq i \leq c - 1$. Note that by definition, if $(u, v) = e \in E(T)$ such that $C(u, v) = c$ and $\Phi(u') = \texttt{black}$ ($\texttt{white}$, respectively), where $u' \in V(T')$, then all the vertices $v'_1, v'_2, \ldots, v'_{c-1}, v \in V(T')$ cannot be colored white (black, respectively).

Therefore, we can replace Problem 1 by the following

Problem 3.

Input: A rooted n-vertex tree $T' = (V', E')$, an integer $d > 1$ and a partial function $\Phi : X \to \{\texttt{black}, \texttt{white}, \texttt{uncolored}\}$, where $X \subset V'$

Output: .

The table root(T').BWU which, for each $\alpha \in \{\texttt{black}, \texttt{white}, \texttt{uncolored}\}$ and $0 \leq x \leq n$, provides the maximal y such that there exists a d-distance anticoloring of T' with root(T') colored α, each $v \in X$ colored $\Phi(v)$ and with x black and y white vertices.

Theorem 3. *Problem 3 is solved in time $O(n^3)$.*

The correctness of Theorem 1 is derived from the above reduction and from the correctness of Theorem 3.

3.2 The Algorithm

The complete proof of Theorem 3 is omitted from this paper. We only give here the algorithm itself with some explanation.

The algorithm implements some dynamic programming techniques. We begin with initializing the lists BWU for each leaf. Then we call a recursive algorithm to find these lists for the internal vertices of the tree (see Algorithm 1). Eventually, we combine the pairs (x, y) of the three arrays root(T).B, root(T).W and root(T).U to a single list, and use a simple procedure, named `contract`, to delete all dominated pairs (as well as repeated occurrences of pairs). This procedure gets a list and uses the bucket-sort algorithm (cf. [4]) to delete from it the dominated pairs, such that eventually we have that each list contains

at most n pairs. The algorithm returns the table dSOL, containing all non-dominated pairs for the root, for which there exists a d-distance anticoloring of the tree.

Algorithm 1 below separates the tree into smaller parts and works recursively on each of them. In fact, the recursion is used only in order to force the calculation to be performed from the leaves to the root of the tree. We deal with the subtrees rooted at each child of a vertex separately and finally merge their results. In general, after the tables for the roots of two subtrees T and T', with a common root but otherwise disjoint sets of vertices, have been generated, the algorithm needs to generate the arrays for $T'' = T \cup T'$. This is done by Algorithm 3.

recAlg(T, r)
Input: A tree T with a root r
Output: r.BWU

if r is a leaf
  return r.BWU
$r_1, r_2, \ldots, r_\delta \leftarrow$ all children of r
for $i \leftarrow 1$ **to** δ
  $r_i.BWU \leftarrow$ recAlg(T, r_i)
  list$_i \leftarrow$ extension(T, r_i.BWU,r) // list for the subtree T_{r_i}, adding r as root
for $i \leftarrow 2$ **to** δ // find the lists for the tree rooted at r
  BWU$_1 \leftarrow$ merge(T,BWU$_1$,BWU$_i$)
r.BWU $\leftarrow$ BWU$_1$
return r.BWU

Algorithm 1: Generate List for the root of a tree

Algorithm 2 makes the lists for the subtree T', created from the subtree T by adding the only outer edge of T's root. The calculation of the array root(T').BWU is divided into three parts, depending on the color of r' =root(T'). Recall that the list $r'.B$ contains the pairs of the solution of the problem for the case r' is black. Therefore, in order to compute such a pair, we have to make sure that the distance from a white vertex is at least d. This check is performed for each pair in $r.U$. The same occurs for $r'.W$. In the third case, where r' =root(T') is uncolored, we have to record in $r'.U[2]$ and $r'.U[3]$ the minimal distance from a black and a white vertex, respectively. Recall that the maximal distance we record is d. We use the procedure `increaseDistance(`d'`)` to increase a distance d' in case $d' < d$. Otherwise, it is not changed. Note that the computation of $r'.U$ is also divided into three parts, depending on the color of r.

Two subtrees T, T' of an n-vertex tree, having the same root, are merged by Algorithm 3. This algorithm gets two lists of pairs, each of size at most n, and outputs the pairwise sums of the pairs in the lists (and sometimes subtracts 1

extension(T, r.BWU,r')
Input: A tree T, the lists r.BWU, where $r = \text{root}(T)$, and the vertex r' =father(r)
Output: r'.BWU – the lists for $r' = \text{root}(T')$, where T' is composed of T and a new root r', whose only child is r

initialize $r'.B, r'.W$ and $r'.U$ to empty lists
$(x_i^b, y_i^b) \leftarrow r.B[i]$ // the i-th pair of $r.B$
$(x_i^w, y_i^w) \leftarrow r.W[i]$ // the i-th pair of $r.W$
$(x_i^u, y_i^u) \leftarrow r.U[1][i]$ // the i-th pair of $r.U[1]$
$r'.B \leftarrow \{(x_i^b + 1, y_i^b) | 1 \leq i \leq n\} \cup \{(x_i^u + 1, y_i^u) \ : \ r.U[3][i] \geq d - 1, 1 \leq i \leq n\}$
// check that the distance from a white vertex is at least d
$r'.W \leftarrow \{(x_i^w, y_i^w + 1) | 1 \leq i \leq n\} \cup \{(x_i^u, y_i^u + 1) \ : \ r.U[2][i] \geq d - 1, 1 \leq i \leq n\}$
$k \leftarrow 1$
for $i = 1$ **to** n // compute $r'.U$
  $r'.U[1][k] \leftarrow (x_i^b, y_i^b)$
  $r'.U[2][k] \leftarrow 1$//distance from a black vertex is 1, since r is black
  $r'.U[3][k] \leftarrow d$//distance from a white vertex must be at least d
  $k \leftarrow k + 1$
  $r'.U[1][k] \leftarrow (x_i^w, y_i^w)$
  $r'.U[2][k] \leftarrow d$//distance from a black vertex is at least d
  $r'.U[3][k] \leftarrow 1$//distance from a white vertex must be 1 since r is white
  $k \leftarrow k + 1$
  $r'.U[1][k] \leftarrow (x_i^u, y_i^u)$
  $r'.U[2][k] \leftarrow$increaseDistance($r.U[2][i]$)//distance from a black vertex
  $r'.U[3][k] \leftarrow$ increaseDistance($r.U[3][i]$)//distance from a white vertex
  $k \leftarrow k + 1$
return $r'.BWU$ //the three calculated arrays

Algorithm 2: Add a new root to a given subtree

from the result). The algorithm computes separately the lists root$(T'').B$, root$(T'').W$ and root$(T'').U$. When computing the distance of root(T'') from a black and a white vertex in T'', we take the minimum distance saved for root(T) and root(T').

3.3 Time Complexity

Obviously, time for the extension algorithm is linear. The time of Algorithm 3 takes $O(n^2)$. This is performed for each vertex of the tree, leading to $O(n^3)$ runtime.

merge(T'', r.BWU,r'.BWU)
Input: A tree T'', the lists r.BWU and r'.BWU, where $r = \text{root}(T)$,
$r' = \text{root}(T')$, $r'' = \text{root}(T'')$ and $T'' = T \cup T'$
Output: r''.BWU – the lists for $r'' = \text{root}(T'')$

$k \leftarrow 1$
for $i = 1$ **to** n //compute $r''.B$
  for $j = 1$ **to** n
    $(x, y) \leftarrow r.B[i]$ //the i-th pair in $r.B$
    $(x', y') \leftarrow r'.B[j]$
    $r''.B[k] \leftarrow (x + x' - 1, y + y')$
    $k \leftarrow k + 1$
for $i = 1$ **to** n //compute $r''.W$
  for $j = 1$ **to** n
    $(x, y) \leftarrow r.W[i]$ //the i-th pair in $r.W$
    $(x', y') \leftarrow r'.W[j]$
    $r''.W[k] \leftarrow (x + x', y + y' - 1)$
    $k \leftarrow k + 1$
for $i = 1$ **to** n //compute $r''.U$
  for $j = 1$ **to** n
    $(x, y) \leftarrow r.U[1][i]$ //the i-th pair in $r.U[1]$
    $(x', y') \leftarrow r'.U[1][j]$
    $r''.U[1][k] \leftarrow (x + x', y + y')$
    $r''.U[2][k] \leftarrow \min\{r.U[2][i], r'.U[2][j]\}$//compute minimal distances
    $r''.U[3][k] \leftarrow \min\{r.U[3][i], r'.U[3][j]\}$
    $k \leftarrow k + 1$
return $r''.BWU$

Algorithm 3: Add a new root to a given subtree

4 Proof of Theorem 2

In this section we describe briefly an algorithm which is given a chordal graph G (here the graph has no weight on its edges), and finds all pairs (x, y) such that there exists a d-distance anticoloring of G with x black and y white vertices.

4.1 Main Idea

Let G be a chordal graph and T a clique tree of G, constructed as in [2]. Note that there is a correspondence between subtrees of T and certain subgraphs of G, whereby for each subtree T' there exists a corresponding subgraph G', induced by $K_1 \cup \ldots \cup K_t$, where $V(T') = \{K_1, K_2, \ldots, K_t\}$.

Before describing the algorithm, let us first present

Definition 4 *[2] Let $G = (V, E)$ be a chordal graph. The `clique graph` of G is denoted by $C(G) = (V_C, E_C, \mu)$, with $\mu : E_C \to \mathbf{N}$, and given by:*

1. *Its vertex set V_C is the set $\{K_1, K_2, \ldots, K_m\}$ of all maximal cliques in G.*
2. *$E_C = \{(K_i, K_j) : K_i, K_j \in V_C,\ K_i \cap K_j \neq \emptyset\}$.*
3. *$\mu(K_i, K_j) = |K_i \cap K_j|$, $(K_i, K_j) \in E_C$.*

Recall that, by [6], a chordal graph contains at most $|V|$ maximal cliques, and therefore $|V_C| \leq |V|$.

Definition 5 *Let $G = (V, E)$ be a chordal graph and $C(G)$ its clique graph. A `clique tree` of G is a maximum weighted spanning tree of $C(G)$.*

Blair *et al.* [2] present a linear time algorithm for finding a clique tree of a chordal graph.

In general, an algorithm which solves the d-distance anticoloring problem on chordal graphs should perform the following steps: First, take a chordal graph G and compute its clique tree T (see Definition 5). Then, perform a similar algorithm to that of the algorithm from Section 3.2 on the clique tree T (described bellow).

Each solution of a clique tree T implies a corresponding d-distance anticoloring for the corresponding chordal graph G. Meaning, each pair (x, y) we find for the clique tree T, implies a d-distance anticoloring of G with x black and y white vertices.

Note that the coloring of the clique tree T need not satisfy the d-distance anticoloring properties, but only that of the 2-distance anticoloring. We allow this since the coloring of T is performed only in order to find the solution for the original graph G.

Similarly to Section 3, in our algorithm each vertex of the tree is attached with three lists, depending on its color, black, white or uncolored. Each list of each vertex contains pairs (x, y) saying that, the graph corresponding to the subtree rooted at that vertex, has a d-distance anticoloring with x black and y

white vertices. Here we use the same definitions for the lists of a vertex v as in Section 3, $v.B, v.W, v.U[1], v.U[2]$ and $v.U[3]$. Similarly to Section 3, these lists are computed in a post-order form, from the leaves of the tree to its root, using two aid procedures: merge and extension.

4.2 A sketch of the Algorithm

Similarly to [10], note that the clique tree T is both vertex- and edge-weighted, where the weight functions $\nu : V(T) \to \mathbf{N}$ and $\mu : E(T) \to \mathbf{N}$ are given by

$$\begin{array}{ll} \nu(K) = |K|, & K \in V(T), \\ \mu(K_1, K_2) = |K_1 \cap K_2|, & (K_1, K_2) \in E(T). \end{array} \tag{1}$$

As in Section 3, our main algorithm uses a dynamic programming techniques to computes the lists for the vertices of the clique tree, from the leaves to the root of the tree. The main algorithm, which is very similar to the algorithm for trees, and is omitted from this paper, invokes Algorithm 4 and 5, which comes instead of Algorithms 2 and 3, respectively. These algorithms need to maintain three lists for each vertex K, named $K.B, K.W$ and $K.U$, where for each pair in $K.U[1]$, we record in $K.U[2]$ ($K.U[3]$, respectively) the distance of K in the coloring represented by $K.U[1]$ from a black (white, respectively) vertex.

Algorithm 4 is similar to Algorithm 2. It is given a clique tree T and the lists for $r =$root(T), and finds the corresponding lists for a new root r', for $r' =$father(r).

In contrast to Algorithm 2, it records for each pair (x, y) belonging to the lists of a vertex K, the furthest colored vertex. This data is saved in `lastColored`, which helps us track the distance we have from a black and a white vertices. These distances are saved in $r.U[2]$ and $r.U[3]$, respectively.

Thus, in case where r is uncolored, it depends on the distance $r.U[2]$ ($r.U[3]$, respectively) whether the new root r' can be colored black (white, respectively).

If r' is also left uncolored, we need to update its distances.

Recall that a clique tree is a maximal spanning tree of the clique graph of G, called $C(G)$. Therefore, there might be some edges in $C(G)$ which are not in T. By the definition of the clique graph [2], there is an edge (r', K) in the clique graph if and only if $r' \cap K \neq \emptyset$. In case where there is an edge between r' and the last colored vertex K in the clique graph $C(G)$, the distance of r' from K is equal to the distance of r from K (see [1],[2],[6],[9]).

Algorithm 5 below is given lists for two subtrees having a common root, and finds the lists for the root of the subtree obtained by merging these two subtrees. For example, for each $(x_1, y_1) \in r_1.B$ and $(x_2, y_2) \in r_2.B$, the algorithm appends to $r.B$ the new pair $(x_1 + x_2 - \texttt{size}, y_1, y_2)$, where `size` is the number of colored vertices in the unified root. Obviously, the colored vertices of $r \subseteq G$ are exactly the colored vertices of $r_1 \cap r_2 \subseteq G$. This data needs to be saved in order to find the coloring for G (finding the coloring itself is omitted from this paper).

Similarly to Algorithm 4, we record here the data for the `lastColored` variable. For each $(x_1, y_1) \in r_1.U$ and $(x_2, y_2) \in r_2.U$, the algorithm appends to

CTextension(T, r.BWU,r')
Input: A clique tree T, the lists r.BWU, where $r = \text{root}(T)$, and $r' = \text{father}(r)$
Output: r'.BWU – the lists for $r' = \text{root}(T')$, where $T' = T \cup \{r'\}$

initialize $r'.B, r'.W$ and $r'.U$ to empty lists
for each $(x, y) \in r.B$
  $(x', y') \leftarrow (x + \nu(r') - \mu(r, r'), y))$
  (x', y').`lastColored` $\leftarrow r'$ and append$(r'.B, (x', y'))$ // add a new pair to $r'.B$
  $(x', y') \leftarrow (x - \mu(r, r'), y - \mu(r, r'))$ // find a pair for $r'.U$
  (x', y').`lastColored` $\leftarrow r$ and append$(r'.U[1], (x', y'))$ // add a new pair to $r'.U$
  $r'.U[2] \leftarrow 1$ // distance from a black vertex is 1
for each $(x, y) \in r.W$
  $(x', y') \leftarrow (x, y + \nu(r') - \mu(r, r'))$
  (x', y').`lastColored` $\leftarrow r'$ and append$(r'.W, (x', y'))$ // add a new pair to $r'.W$
  $(x', y') \leftarrow (x - \mu(r, r'), y - \mu(r, r'))$ // find a pair for $r'.U$
  (x', y').`lastColored` $\leftarrow r$ and append$(r'.U[1], (x', y'))$ // add a new pair to $r'.U$
  $r'.U[3] \leftarrow 1$ // distance from a white vertex is 1
for each $(x, y) \in r.U$ // first check if r' can be colored
  if $r.U[2] \geq d - 1$ // distance from a black vertex
    $(x', y') \leftarrow (x - \mu(r, r'), y + \nu(r') - \mu(r, r'))$
    (x', y').`lastColored` $\leftarrow r'$ and append$(r'.W, (x', y'))$
  if $r.U[3] \geq d - 1$ // distance from a white vertex
    $(x', y') \leftarrow (x + \nu(r') - \mu(r, r'), y - \mu(r, r'))$
    (x', y').`lastColored` $\leftarrow r'$ and append$(r'.B, (x', y'))$
  $(x', y') \leftarrow (x - \mu(r, r'), y - \mu(r, r'))$ // find a pair for $r'.U$
  (x', y').`lastColored` $\leftarrow (x, y)$.`lastColored`
  append$(r'.U[1], (x', y')$
  if the clique graph $C(G)$ contains an edge between r' and (x, y).lastColored
    $r'.U[2] \leftarrow r.U[2]$ and $r'.U[3] \leftarrow r.U[3]$ // distances are not increased
  else //no edge means the distances are increased
    $r'.U[2] \leftarrow r.U[2] + 1$ and $r'.U[3] \leftarrow r.U[3] + 1$
contract$(r'.B)$, contract$(r'.W)$ and contract$(r'.U)$
return $r'.BWU$ //the three calculated arrays

Algorithm 4: Add a new root to a given subtree

$r.U$ the new pair $(x_1 + x_2, y_1 + y_2)$. Here we need to update the distances $r.U[1]$ and $r.U[2]$. We simply choose the minimal distance between the two given ones. The value of `lastColored` will also be the value of the closest colored vertex between the two merged subtrees.

The reduction from a chordal graph to its clique tree and the full algorithm are to be detailed in the full version of the paper.

CTmerge($G, r_1.B, r_2.B, r_1.W, r_2.W$)
Input: $r_i.B, r_i.W$ – the lists for the roots r_i, $i = 1, 2$, of the two subtrees
Output: $r.B, r.W$ – The lists for the unified root r

$r.B, r.W, r.U \leftarrow$ empty lists
for each $(x_1, y_1) \in r_1.B$
  for each $(x_2, y_2) \in r_2.B$
    `size` $\leftarrow |(x_1, y_1).\text{colored} \cup (x_2, y_2).\text{colored}|$ // number of colored vertices
    $(x', y') \leftarrow (x_1 + x_2 - \texttt{size}, y_1 + y_2)$
    $(x', y').$ `lastColored` $\leftarrow r$
    append($r.B, (x', y')$)
    $r.B$.tail.colored $\leftarrow (x_1, y_1).\text{colored} \cap (x_2, y_2).\text{colored}$
for each $(x_1, y_1) \in r_1.W$
  for each $(x_2, y_2) \in r_2.W$
    `size` $\leftarrow |(x_1, y_1).\text{colored} \cup (x_2, y_2).\text{colored}|$
    $(x', y') \leftarrow (x_1 + x_2, y_1 + y_2 - \texttt{size})$
    $(x', y').$ `lastColored` $\leftarrow r$
    append($r.W, (x', y')$)
    $r.W$.tail.colored $\leftarrow (x_1, y_1).\text{colored} \cap (x_2, y_2).\text{colored}$
for each $(x_1, y_1) \in r_1.U[1]$
  for each $(x_2, y_2) \in r_2.U[1]$
    $(x', y') \leftarrow (x_1 + x_2, y_1 + y_2)$
    $r.U[1] \leftarrow$min $\{r_1.U[1], r_2.U[1]\}$ // minimal distance from a black vertex
    $r.U[2] \leftarrow$min $\{r_1.U[2], r_2.U[2]\}$ // minimal distance from a white vertex
    (x', y').lastColored $\leftarrow$ the closest colored vertex
    append($r.W, (x', y')$)
contract($r.B$), contract($r.W$) and contract($r.U$) // delete dominated pairs
return $r.B, r.W, r.U$

Algorithm 5: Merge two subtrees with a common root.

4.3 Runtime of the Algorithm

The construction of a clique graph takes linear time (cf. [2]). Finding a clique tree is done by Prim's algorithm for finding a minimal spanning tree of a graph in $O(|E_C| \lg |V_C|)$ time, where $G_C = (V_C, E_C)$ is the clique graph. The procedure `contract` has a linear runtime.

The algorithm calls Algorithm 4 exactly once for each vertex, and Algorithm 5 exactly d times for each vertex with d children in the clique tree. Thus, it calls both algorithms $O(n)$ times.

Obviously, the runtime of Algorithm 4 is $O(n^2)$.

Each of the double loops in Algorithm 5 is performed over $O(n^2)$ indexes. Each computation of `size` in the loops takes $O(n)$ time. Therefore, the total runtime of Algorithm 5 is $O(n^3)$.

Thus, the total running time of our algorithm is $O(n^4)$.

References

1. P. A. Bernstein and N. Goodman, Power of natural semijoins, *SIAM J. Comput.* 10:751–771, 1981.
2. J. R. S. Blair and B. W. Peyton, An introduction to chordal graphs and clique trees, *Graph Theory and Sparse Matrix Computations*, 56/1-30, Springer Verlag, 1993.
3. N. Bray, from MathWorld–A Wolfram web resource, created by E. W. Weisstein. *url:* mathworld.wolfram.com/GraphStrongProduct.html
4. T. H. Cormen, C. E. Leiserson and R. L. Rivest, Introduction to Algorithms, MIT Press and McGraw-Hill, 1990.
5. P. Galinier, M. Habib and C. Paul, Chordal graphs and their clique graphs, *Graph-Theoretic Concepts in Computer Science*, WG'95, volume 1017 of LNCS, 358–371, 1995
6. F. Gavril, The intersection graphs of subtrees in trees are exactly the chordal graphs, *J. of Comb. Theory*, B/16:47-56, 1974.
7. J. R. Gilbert, D. J. Rose and A. Edenbrandt, A separator theorem for chordal graphs, *SIAM J. Alg. Disc. Meth.* **5** 306–313, 1984.
8. P. Hansen, A. Hertz and N. Quinodoz, Splitting trees, *Disc. Math.*, 165/6:403–419, 1997.
9. M. E. Lundquist, Zero patterns, chordal graphs and matrix completions, PhD thesis, Clemson University, 1990.
10. S. Zucker, The Black-and-White Coloring Problem on Chordal Graphs, *Journal of Graph Algorithms and Applications*, 16/2:261–281, 2012.

Author Index