Editors

Zbigniew Suraj  
Chair of Computer Science  
University of Rzeszow  
Rzeszow  
Poland

Ludwik Czaja  
Institute of Informatics  
University of Warsaw  
and Vistula University  
Warsaw  
Poland

Preface

This two-volume book contains the papers selected for presentation at the Concurrency, Specification and Programming (CS&P) Workshop. It is taking place from 28th to 30th September 2015 in Rzeszow, the biggest city in southeastern Poland.

CS&P provides an international forum for exchanging scientific, research, and technological achievements in concurrency, programming, artificial intelligence, and related fields. In particular, major areas selected for CS&P 2015 include mathematical models of concurrency, data mining and applications, fuzzy computing, logic and probability in theory of computing, rough and granular computing, unconventional computing models. In addition, three plenary keynote talks were delivered.

The Workshop was initiated in the mid-1970s by computer scientists and mathematicians from Warsaw and Humboldt Universities, as Polish-German annual meetings. The first meeting in this series was named 1st Symposium on Mathematical Foundations of Computer Science and it took place in Warsaw from 12th to 19th September 1976. These meetings have been suspended for some years in the eighties until the beginning of ninetieths, and reactivated in 1992. Since then, the Workshop bears the name CS&P, when the first meeting after the break came into effect in Berlin. Now, it is being organized every even year by the Humboldt University of Berlin and every odd year by the University of Warsaw.

It should be mentioned that the CS&P meetings, initially purely bilateral, since 1992 have developed into events attended by participants from a number of various countries beside Poland and Germany. In 2003 the University of Information Technology and Management in Rzeszow, in 2004, the Fraunhofer Institut FIRST in Berlin, and in 2015, the University of Rzeszow joined the organizers as full members of the Committee and financial contributors. The present CS&P 2015 meeting will be hosting participants from the following countries: Canada, Germany, India, Italy, Poland, Russia, Saudi Arabia, Slovakia, Ukraine.

The CS&P 2015 is the twenty-fourth meeting after the break. It received 53 submissions that were carefully reviewed by Program Committee members or external reviewers. After a reviewing process, 49 papers were accepted for presentation at the workshop and publication in the CS&P 2015 proceedings. This book also contains three extended abstracts by the plenary keynote speakers.

It is truly a pleasure to thank all those people who contributed to preparation of this book. In particular, we would like to express our appreciation for the work of the CS&P 2015 Program Committee members and external reviewers who helped to assure the high standards of accepted papers. We would like to thank all the authors of CS&P 2015, without whose high-quality contributions it would not have been possible to organize the workshop. We are grateful to the Organizing Committee members for their involvement in all the organizational matters related to the CS&P 2015 as well as the creation and maintenance of the conference website. We wish to express our thanks to Mikhail Moshkov, Andrzej Skowron and Louchka Popova-Zeugmann for accepting to be plenary speakers at CS&P 2015. We greatly appreciate the financial support received
from the University of Rzeszow, the University of Warsaw, and the Vistula University in Warsaw.

We hope that the CS&P 2015 workshop proceedings will serve as a valuable reference for researchers and developers in the field.

September 2015

Zbigniew Suraj
Ludwik Czaja
Organization

CS&P 2015 was organized by the Chair of Computer Science, the University of Rzeszow, Rzeszow, Poland, in cooperation with the Institute of Mathematics and the Institute of Informatics, the University of Warsaw, Warsaw, Poland, the Vistula University, Warsaw, Poland, the Warsaw Center of Mathematics and Computer Science, Warsaw, Poland, and the Institute of Informatics, the Humboldt University, Berlin, Germany.

CS&P 2015 Conference Committees

Program Committee

- Hans-Dieter Burkhard, Humboldt University, Berlin, Germany
- Ludwik Czaja, University of Warsaw, Warsaw, Poland and Vistula University, Warsaw, Poland
- Monika Heiner, Brandenburg University, Cottbus, Germany
- Anna Gomolinska, University of Bialystok, Białystok, Poland
- Magdalena Kacprzak, Białystok University of Technology, Białystok, Poland
- Hung Son Nguyen, University of Warsaw, Warsaw, Poland
- Wojciech Penczek, Institute of Computer Science, PAS, Warsaw, Poland and University of Podlasie, Siedlce, Poland
- Lech Polkowski, University of Warmia and Mazury, Olsztyn, Poland
- Louchka Popova-Zeugmann, Humboldt University, Berlin, Germany
- Holger Schlingloff, Fraunhofer FIRST and Humboldt University, Berlin, Germany
- Serhat Seker, Vistula University, Warsaw, Poland
- Andrzej Skowron, University of Warsaw, Warsaw, Poland and Systems Research Institute, PAS, Warsaw, Poland
- Zbigniew Suraj, University of Rzeszow, Rzeszow, Poland
- Marcin Szczuka, University of Warsaw, Warsaw, Poland
- Matthias Werner, Technical University, Chemnitz, Germany
- Karsten Wolf, University of Rostock, Rostock, Germany

Organizing Committee

- Aneta Derkacz
- Katarzyna Garwol
- Piotr Grochowalski
- Piotr Lasek
- Łukasz Maciura
- Wiesław Paja
- Krzysztof Pancerz
- Zbigniew Suraj
- Piotr Wasilewski
External Reviewers

- Michal Knapik
- Artur Meski
- Roman Redziejowski
- Anna Sawicka
- Bartłomiej Starosta
- Maciej Szreter
- Dominik Slezak
- Piotr Chrzastowski-Wachtel
Table of Contents

Dynamic Programming Approach for Study of Decision Trees .................. 1
Mikhail Moshkov

Rough Sets in Interactive Granular Computing ................................. 2
Andrzej Skowron

Time and Concurrency - Three Approaches for Intertwining Time and Petri Nets
Louchka Popova-Zeugmann

Comparison of Heuristics for Optimization of Association Rules .............. 4
Fawaz Alsolami, Talha Amin, Mikhail Moshkov, Beata Zielosko

Dynamic Programming Approach for Construction of Association Rule Systems
Fawaz Alsolami, Talha Amin, Igor Chikalov, Mikhail Moshkov, Beata Zielosko

Considering Concurrency in Early Spacecraft Design Studies .................... 22
Jafar Akhundov, Peter Tröger, Matthias Werner

Specialized Predictor for Reaction Systems with Context Properties .......... 31
Roberto Barbuti, Roberta Gori, Francesca Levi, Paolo Milazzo

On Decidability of Persistence Notions ........................................ 44
Kamila Barylska, Lukasz Mikulski

Specifying Functional Programs with Intuitionistic First Order Logic ......... 57
Marcin Benke

Exploration of Knowledge Bases Inspired by Rough Set Theory ............... 64
Agnieszka Nowak-Brzezinska and Alicja Wakulicz-Deja

Complexity Studies for Safe and Fan-Bounded Elementary Hornets .......... 76
Michael Köhler-Bußmeier, Frank Heitmann

Monitoring with Parametrized Extended Life Sequence Charts ................. 88
Ming Chai, Bernd-Holger Schlingloff

Remarks on Memory Consistency Description .................................. 103
Ludwik Czaja

The Method for Describing Changes in the Perception of Stenosis in Blood
Vessels Caused by an Additional Drug ....................................... 115
Sylwia Buregwa-Czuma, Jan G. Bazan, Lech Zareba, Stanisława
Bazan-Socha, Przemysław W. Pardel, Barbara Sokołowska, Łukasz Dydo
Dialogue in Hierarchical Learning of a Concept Using Prototypes and Counterexamples .................................................. 126

Soma Dutta, Piotr Wasilewski

An Approach to Ambiguity Resolution for Ontology Population ............. 134

Natalia Garanina, Elena Sidorova

Application of Genetic Algorithms and High-Performance Computing to the Traffic Signal Setting Problem .................................................. 146

Pawel Gora, Przemyslaw W. Pardel

Lattice Theory for Rough Sets - An Experiment in Mizar ................ 158

Adam Grabowski

Gained and Excluded Classified Actions by Dynamic Security Policies .... 170

Damas P. Gruska

Designing Reliable Communication for Heterogeneous Computer Systems .... 182

Miroslaw Hajder, Janusz Kolbusz, Roman Korostenski

Knowledge Pit - A Data Challenge Platform ........................................ 191

Andrzej Janusz, Dominik Sleczak, Sebastian Stawicki, Mariusz Rosiak

Toward Synchronization of EEG and Eye-tracking Data Using an Expert System 196

Boleslaw Jaskula, Krzysztof Pancerz, Jaroslaw Szkoła

Data Integration through Clustering and Finding Statistical Relations - Validation of Approach .......................................................... 199

Marek Jaszuk, Teresa Mroczek, Barbara Fryc

Selected Methods of Combining Classifiers, when Predictions Are Stored in Probability Vectors, in a Dispersed Decision-Making System .............. 211

Malgorzata Przybyla-Kasperek

Outliers Elimination for Error Correction Algorithm Improvement ........ 223

Janusz Kolbusz, Pawel Rozycki

Core for Large Datasets: Rough Sets on FPGA ..................................... 235

Maciej Kopeczynski, Tomasz Grzes, Jaroslaw Stepaniuk

Sequential P Systems with Active Membranes Working on Sets .......... 247

Michal Kováč, Damas P. Gruska

Author Index ......................................................................................... 258
Dynamic Programming Approach for Study of Decision Trees

Mikhail Moshkov

Computer, Electrical, and Mathematical Sciences and Engineering Division
King Abdullah University of Science and Technology (KAUST)
Saudi Arabia
mikhail.moshkov@kaust.edu.sa

In the presentation, we consider extensions of dynamic programming approach to the study of decision trees as algorithms for problem solving, as a way for knowledge extraction and representation, and as classifiers which, for a new object given by values of conditional attributes, define a value of the decision attribute. These extensions allow us (i) to describe the set of optimal decision trees, (ii) to count the number of these trees, (iii) to make sequential optimization of decision trees relative to different criteria, (iv) to find the set of Pareto optimal points for two criteria, and (v) to describe relationships between two criteria. The results include the minimization of average depth for decision trees sorting eight elements (this question was open since 1968), improvement of upper bounds on the depth of decision trees for diagnosis of 0-1-faults in read-once combinatorial circuits, existence of totally optimal (with minimum depth and minimum number of nodes) decision trees for monotone Boolean functions with at most six variables, study of time-memory tradeoff for decision trees for corner point detection, study of relationships between number and maximum length of decision rules derived from decision trees, study of accuracy-size tradeoff for decision trees which allows us to construct enough small and accurate decision trees for knowledge representation, and decision trees that, as classifiers, outperform often decision trees constructed by CART. The end of the presentation is devoted to the introduction to KAUST.
Rough Sets in Interactive Granular Computing

Andrzej Skowron

Institute of Mathematics, University of Warsaw, Poland
Systems Research Institute, Polish Academy of Sciences
Warsaw, Poland
skowron@mimuw.edu.pl

Decision support in solving problems related to complex systems requires relevant computation models for the agents as well as methods for incorporating reasoning over computations performed by agents. Agents are performing computations on complex objects (e.g., (behavioral) patterns, classifiers, clusters, structural objects, sets of rules, aggregation operations, (approximate) reasoning schemes etc.). In Granular Computing (GC), all such constructed and/or induced objects are called granules. To model, crucial for the complex systems, interactive computations performed by agents, we extend the existing GC approach to Interactive Granular Computing (IGC) by introducing complex granules (c-granules or granules, for short). Many advanced tasks, concerning complex systems may be classified as control tasks performed by agents aiming at achieving the high quality computational trajectories relative to the considered quality measures over the trajectories. Here, new challenges are to develop strategies to control, predict, and bound the behavior of the system. We propose to investigate these challenges using the IGC framework. The reasoning, which aims at controlling the computational schemes from time-to-time, in order to achieve the required targets, is called an adaptive judgement. This reasoning deals with granules and computations over them. Adaptive judgement is more than a mixture of reasoning based on deduction, induction and abduction. Due to the uncertainty the agents generally cannot predict exactly the results of actions (or plans). Moreover, the approximations of the complex vague concepts initiating actions (or plans) are drifting with time. Hence, adaptive strategies for evolving approximations of concepts are needed. In particular, the adaptive judgement is very much needed in the efficiency management of granular computations, carried out by agents, for risk assessment, risk treatment, and cost/benefit analysis. In the lecture, we emphasize the role of the rough set based methods in IGC. The discussed approach is a step towards realization of the Wisdom Technology (WisTech) program, and is developed over years of experiences, based on the work on different real-life projects.
Time and Concurrency - Three Approaches for Intertwining Time and Petri Nets

Louchka Popova-Zeugmann

Department of Computer Science
Humboldt University
Berlin, Germany
popova@informatik.hu-berlin.de

Time and Petri nets - do they not contradict each other? While time determines the occurrences of events in a system, classic Petri nets consider their causal relationships and represent events as a concurrent system. At first, these two appear to be at odds with each other, but taking a closer look at how time and causality are intertwined, one realizes that time actually enriches Petri nets. There are many possible ways in which time and Petri nets interact, this talk will take a short look at three time-dependent Petri nets: Time Petri nets, Timed Petri nets, and Petri nets with time-windows. For the first nets that we will take a look at, Time Petri nets, enabled transitions may fire only during specified time intervals. The transitions must fire the latest at the end of their intervals if they are still enabled then. At any given moment only one transition may fire. This firing does not take time. For the second class of nets, Timed Petri nets, a maximal set of just-enabled transitions fires, and the firing of each transition takes a specific amount of time. The third class of nets, Petri nets with time-windows, portrays time as a minimum and maximum retention for tokens on places. In these nets tokens can be used for firing only during their minimum and maximum retention. At the end of the maximum retention time for a token its time is reset to zero if it was not used for firing. The next period of its retention time on this place then restarts. This repetition can continue indefinitely. For Time Petri nets, we provide an algorithm which proves the behavioral equivalence of a net where time is designed once with real and once with natural numbers. One can also say that the dense semantics of Time Petri nets can be replaced with discrete semantics. For Timed Petri nets, we introduce two time-dependent state equations. These provide a sufficient condition for the non-reachability of states. Last but not least, we prove that Petri nets with time-windows have the ability to realize every transition sequence fired in the net omitting time restrictions. Despite the first experience that time has no influence on the behavior of such nets, we verify that the time can change the liveness behavior of Petri nets with time-windows. We choose these three classes of time-dependent Petri nets to show that time alone does not change the power of a Petri net. In fact, time can or cannot be used to force firing. For Time Petri nets and Timed Petri nets we can say that they are Turing-powerful, and thus more powerful than classic Petri nets. In contrast to these two nets, Petri nets with time-windows have no compulsion to fire. Their expressiveness power is less than that of Turing-machines.
Comparison of Heuristics for Optimization of Association Rules

Fawaz Alsolami\textsuperscript{1}, Talha Amin\textsuperscript{1}, Mikhail Moshkov\textsuperscript{1}, and Beata Zielosko\textsuperscript{2}

\textsuperscript{1} Computer, Electrical and Mathematical Sciences and Engineering Division
King Abdullah University of Science and Technology
Thuwal 23955-6900, Saudi Arabia
{fawaz.alsolami, talha.amin, mikhail.moshkov}@kaust.edu.sa

\textsuperscript{2} Institute of Computer Science, University of Silesia
39, Bedzinska St., 41-200 Sosnowiec, Poland
beata.zielosko@us.edu.pl

Abstract. In this paper, five greedy heuristics for construction of association rules are compared from the point of view of the length and coverage of constructed rules. The obtained rules are compared also with optimal ones constructed by dynamic programming algorithms. The average relative difference between length of rules constructed by the best heuristic and minimum length of rules is at most 4%. The same situation is with coverage.

Key words: greedy heuristics, association rules, decision rules, dynamic programming, rough sets

1 Introduction

Association rule mining is one of the important fields of data mining and knowledge discovery. It aims to extract interesting correlations, associations, or frequent patterns among sets of items in data set.

There are many algorithms for construction of association rules. One of the most popular is Apriori algorithm based on frequent itemsets [1]. During years, many new algorithms were designed which are based on, e.g., hash based technique [15], partitioning the data [18], and others [7, 10, 19].

The most popular measures for mining association rules are support and confidence [9], however in the paper length and coverage as rule evaluation measures are considered. The choice of length is connected with the Minimum Description Length Principle [17]. Shorter rules are better from the point of view of understanding and interpreting by experts. Search of rules with big coverage allows us to discover major patterns in the data, and it is important from the point of view of knowledge representation.

In the paper, greedy algorithms for construction of association rules are studied since the problems of construction of rules with minimum length or maximum coverage are \textit{NP}-hard [6, 12, 14]. The most part of approaches, with the exception of brute-force, Apriori algorithm or extensions of dynamic programming, cannot guarantee the construction of optimal rules (i.e., rules with minimum length or maximum coverage).
In the paper [12], it was shown based on results of U. Feige [8] that, under reasonable assumptions on the class NP, some greedy algorithm is close to the best polynomial approximate algorithms for minimization of association rule length. We do not know about similar results for coverage.

Application of rough sets theory to the construction of rules for knowledge representation or classification tasks are usually connected with the usage of decision table [16] as a form of input data representation. In such a table one attribute is distinguished as a decision attribute and it relates to a rule’s consequence. However, in the last years, associative mechanism of rule construction, where all attributes can occur as premises or consequences of particular rules, is popular. Association rules can be defined in many ways. In the paper, a special kind of association rules is studied, i.e., they relate to decision rules. Similar approach was considered in [12, 13], where a greedy algorithm for minimization of length of association rules was investigated.

In this paper, we consider five greedy heuristics for construction of association rules and compare them from the point of view of the length and coverage of constructed rules. We also compare the obtained rules with optimal ones constructed by dynamic programming algorithms. We show that the average relative difference between length of rules constructed by the best heuristic and minimum length of rules is at most 4%. The same situation is with coverage.

The paper consists of five sections. Section 2 contains main notions. In Sect. 3, we discuss five greedy heuristics. Section 4 contains experimental results for decision tables from UCI Machine Learning Repository, and Sect. 5 – short conclusions.

2 Main Notions

An information system $I$ is a rectangular table with $n+1$ columns labeled with attributes $f_1, \ldots, f_{n+1}$. Rows of this table are filled by nonnegative integers which are interpreted as values of attributes.

An association rule for $I$ is a rule of the kind

$$(f_{i_1} = a_1) \land \ldots \land (f_{i_m} = a_m) \rightarrow f_j = a,$$

where $f_j \in \{f_1, \ldots, f_{n+1}\}$, $f_{i_1}, \ldots, f_{i_m} \in \{f_1, \ldots, f_{n+1}\} \setminus \{f_j\}$, and $a, a_1, \ldots, a_m$ are nonnegative integers.

The notion of an association rule for $I$ is based on the notions of a decision table and decision rule. We consider two kinds of decision tables: with many-valued decisions and with single-valued decisions.

A decision table with many-valued decisions $T$ is a rectangular table with $n$ columns labeled with (conditional) attributes $f_1, \ldots, f_n$. Rows of this table are pairwise different and are filled by nonnegative integers which are interpreted as values of conditional attributes. Each row $r$ is labeled with a finite nonempty set $D(r)$ of nonnegative integers which are interpreted as decisions (values of a decision attribute). For a given row $r$ of $T$, it is necessary to find a decision from the set $D(r)$.

A decision table with single-valued decisions $T$ is a rectangular table with $n$ columns labeled with (conditional) attributes $f_1, \ldots, f_n$. Rows of this table are pairwise different and are filled by nonnegative integers which are interpreted as values of
conditional attributes. Each row $r$ is labeled with a nonnegative integer $d(r)$ which is interpreted as a decision (value of a decision attribute). For a given row $r$ of $T$, it is necessary to find the decision $d(r)$. Decision tables with single-valued decisions can be considered as a special kind of decision tables with many-valued decisions in which $D(r) = \{d(r)\}$ for each row $r$.

For each attribute $f_i \in \{f_1, \ldots, f_{n+1}\}$, the information system $I$ is transformed into a table $I_{f_i}$. The column $f_i$ is removed from $I$ and a table with $n$ columns labeled with attributes $f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_{n+1}$ is obtained. Values of the attribute $f_i$ are attached to the rows of the obtained table $I_{f_i}$ as decisions.

The table $I_{f_i}$ can contain equal rows. We transform this table into two decision tables — with many-valued and single-valued decisions. A decision table $I_{f_i}^{m-v}$ with many-valued decisions is obtained from the table $I_{f_i}$ by replacing each group of equal rows with a single row from the group with the set of decisions attached to all rows from the group. A decision table $I_{f_i}^{s-v}$ with single-valued decisions is obtained from the table $I_{f_i}$ by replacing each group of equal rows with a single row from the group with the most common decision for this group.

The set $\{I_{f_1}^{m-v}, \ldots, I_{f_{n+1}}^{m-v}\}$ of decision tables with many-valued decisions obtained from the information system $I$ is denoted by $\Phi^{m-v}(I)$. We denote by $\Phi^{s-v}(I)$ the set $\{I_{f_1}^{s-v}, \ldots, I_{f_{n+1}}^{s-v}\}$ of decision tables with single-valued decisions obtained from the information system $I$. Since decision tables with single-valued decisions are a special case of decision tables with many-valued decisions, we consider the notion of decision rule for tables with many-valued decisions.

Let $T \in \Phi^{m-v}(I)$. For simplicity, let $T = I_{f_{n+1}}^{m-v}$. The attribute $f_{n+1}$ will be considered as a decision attribute of the table $T$. We denote by $N(T)$ the number of rows in table $T$. For a decision $a$, denote $N(T, a)$ the number of rows $r$ of $T$ such that $a \in D(r)$, and $M(T, a) = N(T) - N(T, a)$. A decision $a$ is a common decision of $T$ if $a \in D(r)$ for any row $r$ of $T$. We denote by $E(T)$ the set of conditional attributes of $T$ which are not constant on $T$. A table obtained from $T$ by removal some rows is called a subtable of $T$. We denote by $T(f_{i_1}, a_1), \ldots, (f_{i_m}, a_m)$ a subtable of $T$ which consists of rows that at the intersection with columns $f_{i_1}, \ldots, f_{i_m}$ have values $a_1, \ldots, a_m$.

The expression

$$(f_{i_1} = a_1) \land \ldots \land (f_{i_m} = a_m) \rightarrow f_{n+1} = a$$

is called a decision rule over $T$ if $f_{i_1}, \ldots, f_{i_m} \in \{f_1, \ldots, f_n\}, a_1, \ldots, a_m$ are the values of the corresponding attributes, and $a$ is a decision. We correspond to the considered rule the subtable $T' = T(f_{i_1}, a_1), \ldots, (f_{i_m}, a_m)$ of the table $T$. This rule is called realizable for a row $r$ of $T$ if $r$ belongs to $T'$. This rule is called true for $T$ if $a$ is a common decision of $T'$. We say that the considered rule is a rule for $T$ and $r$, if this rule is true for $T$ and realizable for $r$. The number $m$ is called the length of the rule. The coverage of the rule is the number of rows $r$ from $T'$ for which $a \in D(r)$. If the considered rule is a rule for $T$ and $r$ then its coverage is equal to $N(T')$.

Decision rules which are true for decision tables from $\Phi^{m-v}(I)$ can be considered as association rules (modification for many-valued decision model) that are true for the information system $I$. Decision rules which are true for decision tables from $\Phi^{s-v}(I)$...
can be considered as association rules (modification for single-valued decision model) that are true for the information system \( I \).

3 Greedy Heuristics

We consider the work of five greedy heuristics on an example of the table \( T = I^{m-v} \).
Let \( r = (b_1, \ldots, b_n) \) be a row of \( T \) and \( a \) be a decision from \( D(r) \). A heuristic \( H \) constructs a decision rule for \( T \) and \( r \). This heuristic starts with a rule whose left-hand side is empty \( \rightarrow f_{n+1} = a \), and then sequentially adds conditions to the left-hand side of this rule. Let during the work of the heuristic \( H \), we already constructed the following rule:

\[
(f_{i_1} = b_{i_1}) \land \ldots \land (f_{i_m} = b_{i_m}) \rightarrow f_{n+1} = a.
\]

We correspond to this rule the subtable \( T' = T(f_{i_1}, b_{i_1}), \ldots, (f_{i_m}, b_{i_m}) \) of the table \( T \). If \( a \) is a common decision for \( T' \) then the work of \( H \) is finished and the constructed rule is returned. Otherwise, we should select a new attribute \( f_{i_{m+1}} \) and construct a new rule:

\[
(f_{i_1} = b_{i_1}) \land \ldots \land (f_{i_m} = b_{i_m}) \land (f_{i_{m+1}} = b_{i_{m+1}}) \rightarrow f_{n+1} = a.
\]

Denote \( T'' = T'(f_{i_{m+1}}, b_{i_{m+1}}), M(f_{i_{m+1}}, r, a) = M(T'', a) = N(T'') - N(T'', a), \) and \( RM(f_{i_{m+1}}, r, a) = (N(T'') - N(T'', a))/N(T'') \). We denote \( \alpha(f_{i_{m+1}}, r, a) = N(T'', a) - N(T'', a) \) and \( \beta(f_{i_{m+1}}, r, a) = M(T', a) - M(T'', a) \). We describe now how five greedy heuristics select the attribute \( f_{i_{m+1}} \).

Heuristic “M” selects an attribute \( f_{i_{m+1}} \in E(T') \) which minimizes the value \( M(f_{i_{m+1}}, r, a) \).

Heuristic “RM” selects an attribute \( f_{i_{m+1}} \in E(T') \) which minimizes the value \( RM(f_{i_{m+1}}, r, a) \).

Heuristic “maxCov” selects an attribute \( f_{i_{m+1}} \in E(T') \) which minimizes the value \( \alpha(f_{i_{m+1}}, r, a) \) given that \( \beta(f_{i_{m+1}}, r, a) > 0 \).

Heuristic “poly” selects an attribute \( f_{i_{m+1}} \in E(T') \) which maximizes the value \( \beta(f_{i_{m+1}}, r, a)/\alpha(f_{i_{m+1}}, r, a)^{\alpha + \beta} \).

Heuristic “log” selects an attribute \( f_{i_{m+1}} \in E(T') \) which maximizes the value \( \beta(f_{i_{m+1}}, r, a)/\log_2[\alpha(f_{i_{m+1}}, r, a) + 2] \).

Let \( H \) be one of the considered heuristics. For a row \( r \) of the table \( T \), we apply it to the row \( r \) and each decision \( a \in D(r) \). As a result, we obtain |\( D(r) \)| rules. Depending on our aim, we either choose among these rules a rule with minimum length or a rule with maximum coverage.

4 Experimental Results

Experiments were made using data sets from UCI Machine Learning Repository [5] and software system Dagger [2]. Some decision rules contain conditional attributes that take unique value for each row. Such attributes were removed. In some tables there were equal rows with, possibly, different decisions. In this case each group of identical
rows was replaced with a single row from the group with the most common decision for this group. In some tables there were missing values. Each such value was replaced with the most common value of the corresponding attribute. Prepared 12 data sets were considered as information systems (see Table 1 which contains some information about each of these information systems).

Table 1. Data sets considered as information systems

<table>
<thead>
<tr>
<th>Data set</th>
<th>Rows</th>
<th>Attr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult-stretch</td>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>625</td>
<td>5</td>
</tr>
<tr>
<td>Breast-cancer</td>
<td>266</td>
<td>10</td>
</tr>
<tr>
<td>Cars</td>
<td>1728</td>
<td>7</td>
</tr>
<tr>
<td>Hayes-roth-data</td>
<td>69</td>
<td>5</td>
</tr>
<tr>
<td>Lenses</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>Monks-1-test</td>
<td>432</td>
<td>7</td>
</tr>
<tr>
<td>Monks-3-test</td>
<td>432</td>
<td>7</td>
</tr>
<tr>
<td>Shuttle-landing</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>Teeth</td>
<td>23</td>
<td>9</td>
</tr>
<tr>
<td>Tic-tac-toe</td>
<td>958</td>
<td>10</td>
</tr>
<tr>
<td>Zoo-data</td>
<td>59</td>
<td>17</td>
</tr>
</tbody>
</table>

For each information system $I$, we construct the set $\Phi^{m-v}(I)$ of decision tables with many-valued decisions and the set $\Phi^{s-v}(I)$ of decision tables with single-valued decisions. For each row $r$ of each table $T \in \Phi^{m-v}(I)$, we apply to this row each of the considered five greedy heuristics as it was described at the end of the previous section. We rank five heuristics for row $r$ relative to the length and coverage of constructed rules and find, for each heuristic, the average ranks relative to length and coverage among all rows of all tables from $\Phi^{m-v}(I)$. After that we consider mean of average ranks among all 12 information systems and obtain overall ranks. Results can be found in Table 2. The best three heuristics for length are M, log, and RM. The best three heuristics for coverage are poly, log, and RM. We study in the same way decision tables with single-valued decisions (see Table 2). The best three heuristics for length are M, RM, and log. The best three heuristics for coverage are poly, log, and RM.

For each heuristic and each row $r$ of each table $T \in \Phi^{m-v}(I)$, we compare the length of rule constructed by heuristic for $r$ (we denote it $length_{\text{greedy}}$) with minimum length of rule (we denote it $length_{\text{min}}$) and calculate the relative difference $\frac{length_{\text{greedy}} - length_{\text{min}}}{length_{\text{min}}}$ (we assume that $0 \div 0 = 0$). The minimum length of rule can be found by dynamic programming algorithms (see [3, 4, 20, 21] for decision tables with single-valued decisions and [11] for decision tables with many-valued decisions). Later, we find average relative difference among all rows of all tables from $\Phi^{m-v}(I)$, and overall average relative difference for all 12 information systems. Results can be found in Table 3. The best three heuristics for the length are M (2% difference), RM (4%), and log (13%). Similar study was done for coverage and decision tables with many-
valued decisions. The relative difference is given by $\frac{\text{coverage}_{\text{max}} - \text{coverage}_{\text{greedy}}}{\text{coverage}_{\text{max}}}$ where $\text{coverage}_{\text{greedy}}$ is the coverage of the rule constructed by greedy heuristic, and $\text{coverage}_{\text{max}}$ is the maximum coverage of the rule calculated by a dynamic programming algorithm. The best three heuristics for the coverage are poly (4% difference), log (8%), and maxCov (14%).

Table 2. Overall ranks for the heuristics

<table>
<thead>
<tr>
<th>Heuristics</th>
<th>poly</th>
<th>log</th>
<th>maxCov</th>
<th>M</th>
<th>RM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-valued decisions</td>
<td>Length</td>
<td>3.38</td>
<td>2.25</td>
<td>5.00</td>
<td>2.17</td>
</tr>
<tr>
<td></td>
<td>Coverage</td>
<td>1.67</td>
<td>1.83</td>
<td>4.00</td>
<td>4.21</td>
</tr>
<tr>
<td>Many-valued decisions</td>
<td>Length</td>
<td>3.33</td>
<td>2.33</td>
<td>5.00</td>
<td>1.79</td>
</tr>
<tr>
<td></td>
<td>Coverage</td>
<td>1.67</td>
<td>1.83</td>
<td>3.67</td>
<td>4.21</td>
</tr>
</tbody>
</table>

We study in the same way decision tables with single-valued decisions (see results in Table 3). The best three heuristics for the length are RM (4% difference), M (5%), and log (14%). The best three heuristics for the coverage are poly (4% difference), log (8%), and maxCov (15%).

Table 3. Overall average relative differences for the heuristics

<table>
<thead>
<tr>
<th>Heuristics</th>
<th>poly</th>
<th>log</th>
<th>maxCov</th>
<th>M</th>
<th>RM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-valued decisions</td>
<td>Length</td>
<td>0.27</td>
<td>0.14</td>
<td>0.84</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>Coverage</td>
<td>0.04</td>
<td>0.08</td>
<td>0.15</td>
<td>0.24</td>
</tr>
<tr>
<td>Many-valued decisions</td>
<td>Length</td>
<td>0.29</td>
<td>0.13</td>
<td>0.83</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>Coverage</td>
<td>0.04</td>
<td>0.08</td>
<td>0.14</td>
<td>0.23</td>
</tr>
</tbody>
</table>

From the considered results it follows that, for the length minimization, we should use the heuristic M and, probably, the heuristic RM. For the coverage maximization we should use the heuristic poly.

5 Conclusions

We compared five heuristics for construction of association rules in the frameworks of both multi-valued and single-valued decision approaches. We shown that the average relative difference between coverage of rules constructed by the best heuristic and maximum coverage of rules is at most 4%. The same situation is with length. In the future, we are planning to use the best heuristic for coverage in algorithms constructing relatively small systems of rules covering almost all objects in information systems.
Acknowledgements

Research reported in this publication was supported by the King Abdullah University of Science and Technology (KAUST).

The authors wish to express their gratitude to anonymous reviewers for useful comments.

References


Dynamic Programming Approach for Construction of Association Rule Systems

Fawaz Alsolami¹, Talha Amin¹, Igor Chikalov¹, Mikhail Moshkov¹, and Beata Zielosko²

¹ Computer, Electrical and Mathematical Sciences and Engineering Division
King Abdullah University of Science and Technology
Thuwal 23955-6900, Saudi Arabia
{fawaz.alsolami, talha.amin, mikhail.moshkov}@kaust.edu.sa
igor.chikalov@gmail.com

² Institute of Computer Science, University of Silesia
39, Bedzinska St., 41-200 Sosnowiec, Poland
beata.zielosko@us.edu.pl

Abstract. In the paper, an application of dynamic programming approach for optimization of association rules from the point of view of knowledge representation is considered. Experimental results present cardinality of the set of association rules constructed for information system and lower bound on minimum possible cardinality of rule set based on the information obtained during algorithm work.

Key words: association rules, decision rules, dynamic programming, set cover problem, rough sets.

1 Introduction

Association rules are popular form of knowledge representation. They are used in various areas such as business field for decision making and effective marketing, sequence-pattern in bioinformatics, medical diagnosis, etc. One of the most popular application of association rules is market basket analysis that finds associations between different items that customers place in their shopping baskets.

There are many approaches for mining association rules. The most popular, is Apriori algorithm based on frequent itemsets [1]. During years, many new algorithms were designed which are based on, e.g., transaction reduction [2], sampling the data [13], and others [7, 9].

The most popular measures for association rules are support and confidence, however in the literature many other measures have been proposed [8, 9]. In this paper, we are interested in the construction of rules which cover many objects. Maximization of the coverage allows us to discover major patterns in the data, and it is important from the point of view of knowledge representation. Unfortunately, the problem of construction of rules with maximum coverage is \emph{NP}-hard [6]. The most part of approaches, with the exception of brute-force and Apriori algorithm, cannot guarantee the construction...
of rules with the maximum coverage. The proposed dynamic programming approach allows one to construct such rules.

Application of rough sets theory to the construction of rules for knowledge representation or classification tasks are usually connected with the usage of decision table [12] as a form of input data representation. In such a table, one attribute is distinguished as a decision attribute and it relates to a rule consequence. However, in the last years, associative mechanism of rule construction, where all attributes can occur as premises or consequences of particular rules, is popular. Association rules can be defined in many ways. In the paper, a special kind of association rules is studied, i.e., they relate to decision rules. Similar approach was considered in [10, 11], where a greedy algorithm for minimization of length of association rules was studied. In [15], a dynamic programming approach to optimization of association rules relative to coverage was investigated.

When association rules for information systems are studied and each attribute is sequentially considered as the decision one, inconsistent tables are often obtained, i.e., tables containing equal rows with different decisions. In the paper, two possibilities of removing inconsistency of decision tables are considered. If in some tables there are equal rows with, possibly, different decisions, then (i) each group of identical rows is replaced with a single row from the group with the most common decision for this group, (ii) each group of identical rows is replaced with a single row from the group with the set of decisions attached to rows from the considered group. In the first case, usual decision tables are obtained (decision tables with single-valued decisions) and, for a given row, we should find decision attached to this row. In the second case, decision tables with many-valued decisions are obtained and, for a given row, we should find an arbitrary decision from the set of decisions attached to this row.

For each decision table obtained from the information system, we construct a system of exact rules in the following way: during each step, we choose a rule which covers the maximum number of previously uncovered rows. We stop the construction when all rows of the table are covered. If the obtained system of rules is short enough, then it can be considered as an intelligible representation of the knowledge extracted from the decision table. Otherwise, we can consider approximate rules, and stop the construction of the rule system when the most part of the rows (for example 90% of the rows) are covered.

In [4], the presented algorithm was proposed as application for multi-stage optimization of decision rules for decision tables. We extend it to association rules. The presented algorithm can be considered as a simulation of a greedy algorithm for construction of partial covers. So we can use lower bound on the minimum cardinality for partial cover based on the information about greedy algorithm work which was obtained in [10].

The paper consists of five sections. Section 2 contains main notions. In Sect. 3, algorithm for construction of system of association rule systems is presented. Section 4 contains experimental results for decision tables from UCI Machine Learning Repository, and Section 5 – short conclusions.
2 Main Notions

An information system $I$ is a rectangular table with $n+1$ columns labeled with attributes $f_1, \ldots, f_{n+1}$. Rows of this table are filled by nonnegative integers which are interpreted as values of attributes.

An association rule for $I$ is a rule of the kind

$$ (f_{i_1} = a_1) \land \ldots \land (f_{i_m} = a_m) \rightarrow f_j = a, $$

where $f_j \in \{f_1, \ldots, f_{n+1}\}$, $f_{i_1}, \ldots, f_{i_m} \in \{f_1, \ldots, f_{n+1}\} \setminus \{f_j\}$, and $a, a_1, \ldots, a_m$ are nonnegative integers.

The notion of an association rule for $I$ is based on the notions of a decision table and decision rule. We consider two kinds of decision tables: with many-valued decisions and with single-valued decisions.

A decision table with many-valued decisions $T$ is a rectangular table with $n$ columns labeled with (conditional) attributes $f_1, \ldots, f_n$. Rows of this table are pairwise different and are filled by nonnegative integers which are interpreted as values of conditional attributes. Each row $r$ is labeled with a finite nonempty set $D(r)$ of nonnegative integers which are interpreted as decisions (values of a decision attribute). For a given row $r$ of $T$, it is necessary to find a decision from the set $D(r)$.

A decision table with single-valued decisions $T$ is a rectangular table with $n$ columns labeled with (conditional) attributes $f_1, \ldots, f_n$. Rows of this table are pairwise different and are filled by nonnegative integers which are interpreted as values of conditional attributes. Each row $r$ is labeled with a nonnegative integer $d(r)$ which is interpreted as a decision (value of a decision attribute). For a given row $r$ of $T$, it is necessary to find the decision $d(r)$. Decision tables with single-valued decisions can be considered as a special kind of decision tables with many-valued decisions in which $D(r) = \{d(r)\}$ for each row $r$.

For each attribute $f_i \in \{f_1, \ldots, f_{n+1}\}$, the information system $I$ is transformed into a table $I_{f_i}$. The column $f_i$ is removed from $I$ and a table with $n$ columns labeled with attributes $f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_{n+1}$ is obtained. Values of the attribute $f_i$ are attached to the rows of the obtained table $I_{f_i}$ as decisions.

The table $I_{f_i}$ can contain equal rows. We transform this table into two decision tables – with many-valued and single-valued decisions. A decision table $I_{f_i}^{m-v}$ with many-valued decisions is obtained from the table $I_{f_i}$ by replacing each group of equal rows with a single row from the group with the set of decisions attached to all rows from the group. A decision table $I_{f_i}^{s-v}$ with single-valued decisions is obtained from the table $I_{f_i}$ by replacing each group of equal rows with a single row from the group with the most common decision for this group.

The set $\{I_{f_1}^{m-v}, \ldots, I_{f_{n+1}}^{m-v}\}$ of decision tables with many-valued decisions obtained from the information system $I$ is denoted by $\Phi^{m-v}(I)$. We denote by $\Phi^{s-v}(I)$ the set $\{I_{f_1}^{s-v}, \ldots, I_{f_{n+1}}^{s-v}\}$ of decision tables with single-valued decisions obtained from the information system $I$. Since decision tables with single-valued decisions are a special case of decision tables with many-valued decisions, we consider the notion of decision rule for tables with many-valued decisions.
Let \( T \in \Phi^{m-v}(I) \). For simplicity, let \( T = I^{m-v}_f \). The attribute \( f_{n+1} \) will be considered as a decision attribute of the table \( T \). We denote by \( \text{Row}(T) \) the set of rows of \( T \). Let \( D(T) = \bigcup_{r \in \text{Row}(T)} D(r) \).

A decision table is called empty if it has no rows. The table \( T \) is called degenerate if it is empty or has a common decision, i.e., \( \bigcap_{r \in \text{Row}(T)} D(r) \neq \emptyset \). We denote by \( N(T) \) the number of rows in the table \( T \) and, for any \( t \in \omega \), we denote by \( N_i(T) \) the number of rows \( r \) of \( T \) such that \( t \in D(r) \). By \( \text{mcd}(T) \) we denote the most common decision for \( T \) which is the minimum decision \( t_0 \) from \( D(T) \) such that \( N_{t_0}(T) = \max \{N_i(T) : t \in D(T)\} \). If \( T \) is empty then \( \text{mcd}(T) = 0 \).

For any conditional attribute \( f_i \in \{f_1, \ldots, f_n\} \), we denote by \( E(T, f_i) \) the set of values of the attribute \( f_i \) in the table \( T \). We denote by \( E(T) \) the set of conditional attributes for which \( |E(T, f_i)| \geq 2 \).

Let \( T \) be a nonempty decision table. A subtable of \( T \) is a table obtained from \( T \) by the removal of some rows. Let \( f_{i_1}, \ldots, f_{i_m} \in \{f_1, \ldots, f_n\} \) and \( a_1, \ldots, a_m \in \omega \) where \( \omega \) is the set of nonnegative integers. We denote by \( T(f_{i_1}, a_1) \ldots (f_{i_m}, a_m) \) the subtable of the table \( T \) containing the rows from \( T \) which at the intersection with the columns \( f_{i_1}, \ldots, f_{i_m} \) have numbers \( a_1, \ldots, a_m \), respectively.

As an uncertainty measure for nonempty decision tables we consider relative misclassification error \( \text{rme}(T) = (N(T) - N_{\text{mcd}(T)}(T))/N(T) \) where \( N_{\text{mcd}(T)}(T) \) is the number of rows \( r \) in \( T \) containing the most common decision for \( T \) in \( D(r) \).

A decision rule over \( T \) is an expression of the kind

\[
(f_{i_1} = a_1) \land \ldots \land (f_{i_m} = a_m) \rightarrow f_{n+1} = t
\]

where \( f_{i_1}, \ldots, f_{i_m} \in \{f_1, \ldots, f_n\} \), and \( a_1, \ldots, a_m, t \) are numbers from \( \omega \). It is possible that \( m = 0 \). For the considered rule, we denote \( T^0 = T \), and if \( m > 0 \) we denote \( T^j = T(f_{i_1}, a_1) \ldots (f_{i_j}, a_j) \) for \( j = 1, \ldots, m \). We will say that the decision rule (1) covers the row \( r = (b_1, \ldots, b_n) \) of \( T \) if \( r \) belongs to \( T^m \), i.e., \( b_{i_j} = a_j \) for \( j = 1, \ldots, m \).

A decision rule (1) over \( T \) is called a decision rule for \( T \) if \( t = \text{mcd}(T^m) \), and if \( m > 0 \), then \( T^{j-1} \) is not degenerate for \( j = 1, \ldots, m \), and \( f_{i_j} \in E(T^{j-1}) \). We denote by \( DR(T) \) the set of decision rules for \( T \).

Let \( \rho \) be a decision rule for \( T \) which is equal to (1). The value \( \text{rme}(T, \rho) = \text{rme}(T^m) \) is called the uncertainty of the rule \( \rho \). Let \( \alpha \) be a real number such that \( 0 \leq \alpha \leq 1 \). We will say that a decision rule \( \rho \) for \( T \) is an \( \alpha \)-decision rule for \( T \) if \( \text{rme}(T, \rho) \leq \alpha \). If \( \alpha = 0 \) (in this case, for each row \( r \) covered by \( \rho \), the set \( D(r) \) contains the decision on the right-hand side of \( \rho \) then we will say that \( \rho \) is an exact rule.

We denote by \( DR_\alpha(T) \) the set of \( \alpha \)-decision rules for \( T \).

### 3 Algorithm for Construction of Association Rule System

\( \alpha \)-Decision rules for tables from \( \Phi^{m-v}(I) \) can be considered as \( \alpha \)-association rules (modification for many-valued decision model) for the information system \( I \). \( \alpha \)-Decision rules for decision tables from \( \Phi^{m-v}(I) \) can be considered as \( \alpha \)-association rules (modification for single-valued decision model) for the information system \( I \). In this section, we consider an algorithm for construction of an association rule system for
One can show that the set of rules for the information system \( I \), a subsystem which is a \( \beta \)-system of \( \alpha \)-decision rules for \( T \) if rules from \( S \) cover at least \((1 - \beta)N(T)\) rows of \( T \).

We describe an algorithm \( \alpha \)-\( \beta \)-Rules which, for a decision table \( T \), and real numbers \( \alpha \) and \( \beta \), constructs a \( \beta \)-system of \( \alpha \)-decision rules for \( T \). During each step, we choose (based on a dynamic programming algorithm [4]) a decision rule which covers maximum number of uncovered previously rows. We stop when the constructed rules cover at least \((1 - \beta)N(T)\) rows of \( T \). We denote by \( \text{Rule}_{\alpha,\beta}(T) \) the constructed system of rules.

We denote by \( C(T, \alpha, \beta) \) the minimum cardinality of a \( \beta \)-system of \( \alpha \)-decision rules for \( T \). It is clear that \( C(T, \alpha, \beta) \leq |\text{Rule}_{\alpha,\beta}(T)| \). Using information based on the work of algorithm \( \alpha \)-\( \beta \)-Rules, we can obtain lower bound on the parameter \( C(T, \alpha, \beta) \).

During the construction of \( \beta \)-system of \( \alpha \)-decision rules for \( T \), let the algorithm \( \alpha \)-\( \beta \)-Rules selects consequently rules \( \rho_1, \ldots, \rho_t \). Let \( B_1, \ldots, B_t \) be sets of rows of \( T \) covered by rules \( \rho_1, \ldots, \rho_t \), respectively. Set \( B_0 = \emptyset \) and, for \( i = 1, \ldots, t \), set \( \delta_i = |B_i \setminus (B_0 \cup \ldots \cup B_{i-1})| \). The information derived from the algorithm’s work consists of the tuple \((\delta_1, \ldots, \delta_t)\) and the numbers \( N(T) \) and \( \beta \).

From the results obtained in [10] regarding a greedy algorithm for the set cover problem it follows that \( C(T, \alpha, \beta) \geq l(T, \alpha, \beta) \) where

\[
l(T, \alpha, \beta) = \max \left\{ \left( \frac{[(1 - \beta)N(T)] - (\delta_0 + \ldots + \delta_i)}{\delta_{i+1}} \right) : i = 0, \ldots, t - 1 \right\}.
\]

Using algorithm \( \alpha \)-\( \beta \)-Rules, for each decision table \( T \in \Phi^{m-v}(I) \), we construct the set of rules \( \text{Rule}_{\alpha,\beta}(T) \). As a result, we obtain the system of rules (\( \alpha \)-association rules for the information system \( I \) – modification for many-valued decision model) \( \text{Rule}^{m-v}_{\alpha,\beta}(I) = \bigcup_{T \in \Phi^{m-v}(I)} \text{Rule}_{\alpha,\beta}(T) \). This system contains, for each \( T \in \Phi^{m-v}(I) \), a subsystem which is a \( \beta \)-system of \( \alpha \)-decision rules for \( T \). We denote by \( C^{m-v}(I, \alpha, \beta) \) the minimum cardinality of such system. One can show that

\[
L^{m-v}(I, \alpha, \beta) \leq C^{m-v}(I, \alpha, \beta) \leq U^{m-v}(I, \alpha, \beta),
\]

\[
L^{m-v}(I, \alpha, \beta) = \sum_{T \in \Phi^{m-v}(I)} l(T, \alpha, \beta) \text{ and } U^{m-v}(I, \alpha, \beta) = \left| \text{Rule}^{m-v}_{\alpha,\beta}(I) \right|.
\]

We can do the same for the set \( \Phi^{s-v}(I) \) of decision tables with single-valued decisions. As a result, we obtain the system of rules (\( \alpha \)-association rules for the information system \( I \) – modification for single-valued decision model) \( \text{Rule}^{s-v}_{\alpha,\beta}(I) = \bigcup_{T \in \Phi^{s-v}(I)} \text{Rule}_{\alpha,\beta}(T) \) which contains, for each \( T \in \Phi^{s-v}(I) \), a subsystem which is a \( \beta \)-system of \( \alpha \)-decision rules for \( T \). Denote \( C^{s-v}(I, \alpha, \beta) \) the minimum cardinality of such system. One can show that

\[
L^{s-v}(I, \alpha, \beta) \leq C^{s-v}(I, \alpha, \beta) \leq U^{s-v}(I, \alpha, \beta),
\]
### Table 1. Total number of rules (upper bound / lower bound)

<table>
<thead>
<tr>
<th>Information system</th>
<th>Rows</th>
<th>Attr</th>
<th>( \alpha = 0, \beta = 0 )</th>
<th>( \alpha = 0.3, \beta = 0.2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>adult-stretch</td>
<td>16</td>
<td>5</td>
<td>( 16^{2} )</td>
<td>( 16^{2} )</td>
</tr>
<tr>
<td>balance-scale</td>
<td>625</td>
<td>5</td>
<td>( 570 )</td>
<td>( 570 )</td>
</tr>
<tr>
<td>breast-cancer</td>
<td>266</td>
<td>10</td>
<td>( 984 )</td>
<td>( 984 )</td>
</tr>
<tr>
<td>cars</td>
<td>1728</td>
<td>7</td>
<td>( 358 )</td>
<td>( 358 )</td>
</tr>
<tr>
<td>hayes-roth-data</td>
<td>69</td>
<td>5</td>
<td>( 200 )</td>
<td>( 200 )</td>
</tr>
<tr>
<td>lenses</td>
<td>24</td>
<td>5</td>
<td>( 16 )</td>
<td>( 16 )</td>
</tr>
<tr>
<td>monks-1-test</td>
<td>432</td>
<td>7</td>
<td>( 36 )</td>
<td>( 36 )</td>
</tr>
<tr>
<td>monks-3-test</td>
<td>432</td>
<td>7</td>
<td>( 36 )</td>
<td>( 36 )</td>
</tr>
<tr>
<td>shuttle-landing</td>
<td>15</td>
<td>7</td>
<td>( 22 )</td>
<td>( 22 )</td>
</tr>
<tr>
<td>teeth</td>
<td>23</td>
<td>9</td>
<td>( 86 )</td>
<td>( 86 )</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>958</td>
<td>10</td>
<td>( 512 )</td>
<td>( 512 )</td>
</tr>
<tr>
<td>zoo-data</td>
<td>59</td>
<td>17</td>
<td>( 76 )</td>
<td>( 76 )</td>
</tr>
</tbody>
</table>

\[
L^{s-v}(I, \alpha, \beta) = \sum_{T \in \Phi^{s-v}(I)} l(T, \alpha, \beta) \quad \text{and} \quad U^{s-v}(I, \alpha, \beta) = \left| \text{Rule}_{\alpha, \beta}^{s-v}(I) \right|.
\]

### 4 Experimental Results

Experiments were made using data sets from UCI Machine Learning Repository [5] and software system Dagger [3]. Some decision tables contain conditional attributes that take unique value for each row. Such attributes were removed. In some tables, there were equal rows with, possibly, different decisions. In this case each group of identical rows was replaced with a single row from the group with the most common decision for this group. In some tables there were missing values. Each such value was replaced with the most common value of the corresponding attribute.

Prepared 12 data sets were considered as information systems. Table 1 contains name (column “Information system”), number of rows (column “Rows”), and number of attributes (column “Attr”) for each of the considered information systems. Table 1 presents also upper / lower bounds (see descriptions at the end of the previous section) on \( C^{m-v}(I, \alpha, \beta) \) (column “many-val”) and on \( C^{s-v}(I, \alpha, \beta) \) (column “single-val”) for pairs \((\alpha = 0, \beta = 0)\) and \((\alpha = 0.3, \beta = 0.2)\).

We can see that, for tables with many-valued decisions, upper and lower bounds on the number of rules are less than or equal to the bounds for decision tables with single-valued decision. We considered a threshold

\[
30 \times (\text{number of attributes})
\]
Table 2. Total number of rules for information system balance-scale with 5 attributes

<table>
<thead>
<tr>
<th>α \ β</th>
<th>0</th>
<th>0.1</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>847</td>
<td>793</td>
<td>670</td>
</tr>
<tr>
<td>0.01</td>
<td>633</td>
<td>695</td>
<td>651</td>
</tr>
<tr>
<td>0.05</td>
<td>772</td>
<td>698</td>
<td>595</td>
</tr>
<tr>
<td>0.1</td>
<td>603</td>
<td>553</td>
<td>516</td>
</tr>
<tr>
<td>0.15</td>
<td>614</td>
<td>510</td>
<td>447</td>
</tr>
<tr>
<td>0.2</td>
<td>534</td>
<td>444</td>
<td>390</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>α \ β</th>
<th>0</th>
<th>0.1</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>507</td>
<td>373</td>
<td>330</td>
</tr>
<tr>
<td>0.01</td>
<td>493</td>
<td>358</td>
<td>317</td>
</tr>
<tr>
<td>0.05</td>
<td>432</td>
<td>298</td>
<td>255</td>
</tr>
<tr>
<td>0.1</td>
<td>353</td>
<td>219</td>
<td>180</td>
</tr>
<tr>
<td>0.15</td>
<td>374</td>
<td>163</td>
<td>107</td>
</tr>
<tr>
<td>0.2</td>
<td>194</td>
<td>108</td>
<td>68</td>
</tr>
</tbody>
</table>

as a reasonable upper bound on the number of rules if a system of rules is used for knowledge representation. In the case $\alpha = 0$ and $\beta = 0$, the threshold is exceeded for five information systems (see numbers in bold): balance-scale, breast-cancer, cars, hayes-roth-data, and tic-tac-toe. The consideration of approximate rules and partial covers can improve the situation. In the case $\alpha = 0.3$ and $\beta = 0.2$, the threshold is exceeded for three information systems (balance-scale, breast-cancer, and tic-tac-toe) if we consider decision tables with single-valued decisions and for two information systems (breast-cancer and tic-tac-toe) if we consider decision tables with many-valued decisions.

For four information systems (balance-scale, breast-cancer, cars, and hayes-roth-data), upper / lower bounds on $C^{m-v}(I, \alpha, \beta)$ and on $C^{s-v}(I, \alpha, \beta)$ for $\beta \in \{0, 0.01, 0.05, 0.1, 0.15, 0.2\}$ and $\alpha \in \{0, 0.1, 0.3\}$ can be found in Tables 2, 3, 4, and 5.

5 Conclusions

In the paper, an algorithm for construction of association rule system is proposed. It simulates the work of greedy algorithm for set cover problem. Experimental results present cardinality of the set of association rules constructed for information system and lower bound on minimum possible cardinality of such set based on the information about the algorithm work. In the future, the length of constructed association rules will be studied also. We are planning to extend an approach proposed in [14] for decision rules to construction of association rule systems. This approach allows one to construct rules with coverage close to maximum and requires less time than the dynamic programming approach.
Table 3. Total number of rules for information system breast-cancer with 10 attributes

<table>
<thead>
<tr>
<th>α</th>
<th>0</th>
<th>0.1</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1027</td>
<td>570</td>
<td>559</td>
</tr>
<tr>
<td>0.01</td>
<td>1007</td>
<td>556</td>
<td>546</td>
</tr>
<tr>
<td>0.05</td>
<td>905</td>
<td>496</td>
<td>485</td>
</tr>
<tr>
<td>0.1</td>
<td>784</td>
<td>426</td>
<td>418</td>
</tr>
<tr>
<td>0.15</td>
<td>684</td>
<td>369</td>
<td>361</td>
</tr>
<tr>
<td>0.2</td>
<td>596</td>
<td>321</td>
<td>314</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>α</th>
<th>0</th>
<th>0.1</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>984</td>
<td>522</td>
<td>512</td>
</tr>
<tr>
<td>0.01</td>
<td>964</td>
<td>506</td>
<td>498</td>
</tr>
<tr>
<td>0.05</td>
<td>862</td>
<td>446</td>
<td>438</td>
</tr>
<tr>
<td>0.1</td>
<td>741</td>
<td>383</td>
<td>376</td>
</tr>
<tr>
<td>0.15</td>
<td>645</td>
<td>335</td>
<td>327</td>
</tr>
<tr>
<td>0.2</td>
<td>562</td>
<td>295</td>
<td>288</td>
</tr>
</tbody>
</table>

Table 4. Total number of rules for information system cars with 7 attributes

<table>
<thead>
<tr>
<th>α</th>
<th>0</th>
<th>0.1</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>114</td>
<td>237</td>
<td>235</td>
</tr>
<tr>
<td>0.01</td>
<td>108</td>
<td>213</td>
<td>212</td>
</tr>
<tr>
<td>0.05</td>
<td>956</td>
<td>136</td>
<td>135</td>
</tr>
<tr>
<td>0.1</td>
<td>822</td>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td>0.15</td>
<td>741</td>
<td>72</td>
<td>72</td>
</tr>
<tr>
<td>0.2</td>
<td>64</td>
<td>42</td>
<td>42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>α</th>
<th>0</th>
<th>0.1</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>158</td>
<td>203</td>
<td>203</td>
</tr>
<tr>
<td>0.01</td>
<td>150</td>
<td>183</td>
<td>183</td>
</tr>
<tr>
<td>0.05</td>
<td>118</td>
<td>112</td>
<td>112</td>
</tr>
<tr>
<td>0.1</td>
<td>124</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>0.15</td>
<td>69</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>0.2</td>
<td>64</td>
<td>34</td>
<td>34</td>
</tr>
</tbody>
</table>
Table 5. Total number of rules for information system hayes-roth-data with 5 attributes

<table>
<thead>
<tr>
<th>α</th>
<th>0</th>
<th>0.1</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>169</td>
<td>169</td>
<td>169</td>
</tr>
<tr>
<td>0.01</td>
<td>169</td>
<td>169</td>
<td>169</td>
</tr>
<tr>
<td>0.05</td>
<td>158</td>
<td>158</td>
<td>158</td>
</tr>
<tr>
<td>0.1</td>
<td>143</td>
<td>143</td>
<td>143</td>
</tr>
<tr>
<td>0.15</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>0.2</td>
<td>115</td>
<td>115</td>
<td>115</td>
</tr>
</tbody>
</table>

Acknowledgements

Research reported in this publication was supported by the King Abdullah University of Science and Technology (KAUST).

The authors wish to express their gratitude to anonymous reviewers for useful comments.

References

Abstract. In real-world spacecraft systems, concurrent system activities must be constrained for energy efficiency and functional reasons. Such constraints must be considered in the early design phases, in order to avoid costly reiterations and modifications of the proposed system design in later phases. Although some initial attempts for using formal specifications exist in the domain, there is a lack of concurrency support in the utilized approaches. In this paper, we therefore first formalize an existing domain-specific language for specifying spacecraft designs and their constraints. Since this language does not support the modelling of concurrency issues, we extend it accordingly and map it to standard timed automata, based on a general system model. The new-style specifications can now be processed with existing and proven automata modelling tools, which enables faster and more reliable feasibility checks for early spacecraft system designs.

1 Introduction

The model-based engineering paradigm is widely accepted in the aerospace industry. One specific implementation is "Concurrent Engineering (CE)", an approach that has evolved into an important support mechanism for early space mission design decisions. It is supposed to clarify the baseline requirements and design properties for a future spacecraft, such as overall mass, power consumption and costs for development and operation. Studies have shown that up to 85% of the overall spacecraft project costs result from decisions made in this early phase [14].

During the CE project phase, multiple spacecraft design alternatives are simultaneously created, refined and analyzed by a group of engineers. Everybody in this group has an own range of responsibilities, such as the consideration of satellite trajectories, the realization of communication mechanisms, or the consideration of thermal aspects [11]. This makes the evaluation and ranking of architecture variations a complicated issue, since local design changes always have implications on global system scope. Those modifications may even impact the feasibility of the complete system design, if not correctly tracked and understood before the next project phase. This led to the development of feasibility checking approaches as part of CE in the past. They are conducted iteratively after each major design change, to provide feedback for engineers and the spacecraft customer.

The following paper discusses the consideration of concurrency demands and restrictions in the CE project phase based on a formal specification. Section 2 first introduces an abstract system model for spacecraft systems, in order to restrict the model
entities to a necessary minimum. Section 3 then explains a domain-specific language being used in a German spacecraft project for the formulation of constraints and properties. In Section 4, we show how this DSL must be enhanced to consider concurrency issues properly. Section 5 finally explains how the resulting DSL can be translated to timed automata for better analysis support from existing tools.

2 Spacecraft System Model

A spacecraft system can be modeled as a set of internal and external state variables $S = IS \cup ES$. The internal state $si$ is defined by all state variables that can be directly manipulated through spacecraft’s activities, e.g., the charging level of the battery or the available free memory. External state variables change by themselves and are not directly affected by spacecraft activities. Examples are temperature, the time, or the availability of sun light. They constitute the external state $se$.

The spacecraft activity can be abstractly defined based on the idea of interacting components. One relevant property here is the consideration of their concurrent activities and their relation to time. Let $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}, n \in \mathbb{N}$ be a set of operational states representing activities of the spacecraft components, e.g. the active charging of the satellite battery from solar panels or the usage of some data transmission down-link. I.e., for each $\tau_i$, there exists a transition function $f_i(se, si, \Delta t) = \Delta si$ that describes the change of the internal system’s state if a component is active for the time $\Delta t$.

In real-world satellite systems, concurrent component activities must be constrained for energy efficiency and functional reasons. Not all spacecraft components can be active at the same time. Let $\mathcal{C} = \{c_1, \ldots, c_k\}, k \in \mathbb{N}$ represent a set of constraints for the state variables. Please note that the mission goals can be seen as special mission constraints.

Given the above definitions, a schedule or a plan is defined as a mapping of the component activity set $\mathcal{T}$ onto a set of time intervals:

$$\mathcal{P} : \mathcal{T} \mapsto \mathcal{I},$$

where $\mathcal{I} = \{(b, f) : b, f \in \mathbb{R}^+, b < f\}$ is a set of finite length time intervals with $b, f$ begin the start- and end-times of the intervals, respectively. As a side condition, all $C_i$ must be true at any time.

Based on the given system model, we are now discussing how constraints can be formulated and checked for a specific system model instance.

3 DSL-Based Constraint Description

Different methods can be used to check a spacecraft system model’s compliance with the intended constraints and goals for the planned mission time. Schaus et al. described how an expert can formulate such requirements in a dedicated domain specific language (DSL) and use search algorithms to check the feasibility with respect to the given constraints [10]. The proposed syntax was used to create a description for a representative flying mission called satellite TET-1 [7, 16]. The DSL syntax turned out to be detailed
enough to be fed into a search algorithm that looks for an execution trace fulfilling all conditions as mission plan [12].

Originally, the proposed concept was called continuous verification. We avoid the term ‘verification’ intentionally here, since at such an early design phase there is no real detailed system specification against which the system would be formally verified. Actually, creating unambiguous specifications is the purpose of the whole preliminary analysis. Instead of ‘verification’, the terms ‘feasibility study/check’ will be used.

In the published DSL syntax, the engineer can define different system components being exclusively active (called states) and system parameters (called operational state variables) which can be changed by the components’ activities. The values of these parameters define the global system state. I.e., there exist implicit constraints

\[ \forall i, j \neq j, \tau_i \Rightarrow \neg \tau_j \]  

(2)

The mission goal is defined as a constraint on the operational state variables. Additional constraints beside the central mission goal can also be formulated. Furthermore, for every component there are corresponding rates of changes to the system parameters. The generated mission plan is a chain of activity periods derived from this description, such as Battery Charge \( \rightarrow \) Collect Data with Cameras \( \rightarrow \) Send Data to Earth \( \rightarrow \) Idle \( \rightarrow \) ....

Since satellite systems depend not only on the internal components, but also on the execution environment, there is a possibility to describe such environmental properties in the DSL, too. This is done by parameterizing external simulation models (SGP4) [10, 6] to compute satellite trajectories and generate periodic external events that activate components.

The original DSL from [10] is translated to the grammar description in Figure 1. It contains of the following parts:

- The solver construct determines the actual search algorithm which is used to find an appropriate solution.
- The state parameters together with the initial parameter values define system parameters which determine spacecraft system’s state.
- The operational states represent the components of the satellite being active.
- The starting state defines the first component being active.
- The operational state changes define rates of change of system parameters by components’ activities.
- For some of the state parameters, there could exist upper or lower bounds that are determined in the operational constraints construct.
- The goal of the whole mission can be defined as a set of state parameter values which can be set in the operational goals section.
- External events and intervals are determined by using external simulation modules being defined. Only the construct for a simulation step is given, since the rest is initialization of the satellite’s trajectory and coordinates of the ground stations for down-link and up-link communications.

Since in the original language there is no component concurrency being considered, only one component resp. operational state can be active at any point in time. Consequently, there is no explicit time notion in the language, with the only exception of
Fig. 1. Syntactical structure of an existing domain-specific language for spacecraft design descriptions, derived from [10]. Some non-terminals, such as identifiers, are omitted due to space restrictions.
simulation step which is the parameter fed to the external orbit simulator. This also de-
termines the values for the changes of state parameters and the maximum number of
steps before the mission deadline.

An important issue in the aforementioned DSL is that the operational state of the
system, a vector of all state variables, is assumed to be directly bounded to the state
of the currently active component. In other words: The system is specified with having
only one active component at a time. This is an obvious over-constraining of the model,
since a satellite as a system cannot be observed isolated from its environment. Concur-
rency consideration should be therefore not only a possible, but a desirable property.
Several components of a satellite can be active simultaneously. One simple example is
data being sent to earth while the battery is being charged. Furthermore, such a restric-
tive execution model can lead to the false negative check result, although the mission
could still be possible to execute with a more “dense” plan

Supporting concurrency in the DSL and the implied execution model would lead to
increased expressiveness of the language and allow for more realistic derivable analysis
results. Furthermore, with the concurrent execution model the generated plans are more
‘dense’ and thus the probability of a false negative result of the feasibility checking is
lower.

4 Extended DSL Syntax

Given the formal system model from Section 2, we translate the semantics of the given
DSL first into the set-theoretical constructs to understand the missing expressivity and
determine the necessary enhancements:

– A solver is implementation-specific and defines a search method to trace the state
  space of a given problem. Hence, it is not part of the formal model.
– The state parameters are represented by the set $S$ of state variables in the formal
  model. Their initial values are implementation-specific.
– The operational states are defined by the set $T$ of the formal model. Starting State
does not need to be determined by the engineer since a satellite can have several
active components simultaneously and their enabling depends on the validity con-
ditions.
– The operational state changes are also defined by the set $T$ of the formal model.
– The operational goals and constraints are represented by the set $C$ of the formal
  model.

There is no explicit notion of duration in the given DSL. The set of prohibited overlapped
component activities is also missing in the explicit form. Implicitly, however, one
could say that the set is given by the assumption in the execution model that all overlapped
executions are forbidden. Furthermore, events which enable certain component
activities are not explicitly present in the DSL. They are generated by the simulation
modules and directly fed to the search algorithm.

1 Here, the term “plan density” is loosely coined to describe a possibility to use overlapping
time intervals to pack more activity into the mission time.
For supporting the missing concurrency semantics, the DSL is now enhanced with three syntactical constructs, as shown in Figure 2: Explicit durations per component activity, prohibited component activity overlapping, and events + durations.

Fig. 2. DSL Enhancements for Concurrency Support

As a result of the reformulation and enhancement of the DSL grammar, the problem of feasibility analysis can now be reformulated as an off-line scheduling problem with several constraints [8]. Therefore, any formal modeling method supporting concurrency and having explicit notion of time would be a potential solution helper for the problem. Well-known examples are Timed CSP, Timed Petri Nets, Timed Automata, or Hybrid Automata.

We decided to map the enhanced DSL to Timed Automata [1, 2] for several reasons:

- The concepts of timed automata are easier accepted by engineers unrelated to computational logic and formal modeling.
- The structure of the DSL introduced in [10] translates one-to-one to a automata description.
- Timed automata have been proven to be more expressive than Timed Petri Nets [5] and just as expressive as Timed CSP models[9]

Although hybrid automata are more general and more expressive than timed automata [13], we sticked with pure time automata eventually, since they provide enough modeling capabilities for the given concurrency analysis problem.
5 Determining Feasibility

5.1 Timed Automata Representation

Mapping the enhanced DSL to timed automata constructs enables the system modeler to generate mission plans with appropriate existing tools, based on a transformed description with the DSL. Generally, the following definition holds:

Definition 1. A timed automaton $A$ is a tuple $(L, l_0, X, Inv, T, \Sigma)$ where:

- $L$ is a finite set of states (locations)
- $l_0$ is the initial state
- $X$ is a finite set of synchronously running real-valued clocks
- $T \subseteq L \times C(X) \times \Sigma \times 2^X \times L$ is a finite set of transitions $l \xrightarrow{g,a,r} l'$, where $g$ is the guarding condition for the transition, $a$ is the action and $r$ is a set of clocks which are reset by the transition
- $Inv : L \rightarrow C(X)$ represents an invariant for every corresponding location
- $\Sigma$ is an alphabet of all possible actions.

Timed automata can model communication through binary synchronization, represented here by synchronization actions that are represented by according transitions. They can be either input actions ($\text{sync?}$) or output actions ($\text{sync!}$) that are performed in pair. [15, 3]

In the sequential execution model of the original DSL, only one operational state was allowed to be active at a time. Hence, in such a representation, the whole set of components can be modeled as a single automaton with single inactive state (for idle mode) and several active states - one for each component. The values of system parameters will change with the given rates in some finite fixed step size which will be measured by a global clock.

Mapping the enhanced DSL description to the timed automata formalism is now straightforward. An operational state or component activity can be represented by a single automaton consisting of two locations - Active and Inactive. Validity conditions for the component to become active are represented by guards in the language of the TA. Clocks represent a maximum duration, if any is given. A component can remain active or inactive so long as the invariants are correspondingly fulfilled.

Mission goals and constraints can be embedded in two ways. Constraints are simply the guards on the transitions or location invariants which cannot be violated. To each goal there exists a dedicated automaton which also consists of two states. Once the goal is reached this automaton moves from one location to the the second and remains there representing a fulfilled goal. If all of the goals are satisfied before the global clock reaches some predefined value (mission time constraints), then the mission is feasible.

5.2 Application in Case Study

Based on the original DSL expressiveness, [12] researched implementations of various search methods, heuristics and optimizations of verification run times for different kinds of space missions. Depending on the complexity of the system and on the chosen
technique, execution times varied from fractions of a second up to several minutes on an i7 running at 3 Ghz with 8 Gb of RAM. The main challenge of [12] has been to find an optimal value function and heuristics to minimize the effects of state space explosion. However, the system schedule did not make any statement beyond satisfiability of mission constraints and mission feasibility for the case where system components are activated in a sequential manner.

Using the proposed enhanced DSL and its mapping to an automata formalism, the TET-1 mission was again modelled by the authors in the UPPAAL tool [3, 4]. Specifically, it allowed us to perform a reachability analysis and to create a simulation trace as representation of a feasible mission plan with concurrent component activities.

We investigated a major improvement of the analysis run-time in comparison to the original behavior, which is shown in Table 1. Feasibility check times are now in the sub-second area. In comparison to the sequential methods used in [10], this introduces an improvement of almost two orders of magnitude of run-time with a more realistic, concurrent execution model.

Table 1. Performance of the feasibility check with UPPAAL 4.0.18. The average results are given for three different search methods available in the tool.

<table>
<thead>
<tr>
<th>Search Order</th>
<th>10 Gb Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breadth first</td>
<td>out of memory error</td>
</tr>
<tr>
<td>Depth first</td>
<td>0.35 s</td>
</tr>
<tr>
<td>Randomized Depth first</td>
<td>0.34 s</td>
</tr>
</tbody>
</table>

6 Summary

In the earliest design phases of spacecraft systems, concurrent of component activities must already be considered to create optimized but still feasible system designs and mission plans. The article discussed a specific domain specific language being used for describing such system designs and their constraints. We showed how the expressiveness of the DSL can be extended to describe concurrent system activities. We furthermore presented a mapping to standard timed automata, which allows the re-use the broad existing body of tools for evaluation and analysis. Our enhanced description methodology has shown to be applicable for real-world space mission systems, extending the state of the art in early spacecraft design methodologies.

References

Specialized Predictor for Reaction Systems with Context Properties

Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo
Dipartimento di Informatica, Università di Pisa, Italy

Abstract. Reaction systems are a qualitative formalism for modeling systems of biochemical reactions characterized by the non-permanency of the elements: molecules disappear if not produced by any enabled reaction. Reaction systems execute in an environment that provides new molecules at each step. Brijder, Ehrenfeucht and Rozemberg introduced the idea of predictors. A predictor of a molecule $s$, for a given $n$, is the set of molecules to be observed in the environment in order to determine whether $s$ is produced or not by the system at step $n$. We introduced the notion of formula based predictor, that is a propositional logic formula that precisely characterizes environments that lead to the production of $s$ after $n$ steps. In this paper we revise the notion of formula based predictor by defining a specialized version that assumes the environment to provide molecules according to what expressed by a temporal logic formula. As an application, we use specialized formula based predictors to give theoretical grounds to previously obtained results on a model of gene regulation.

1 Introduction

In the context of Natural Computing [11, 6], reaction systems [10, 2] have been proposed as a model for the description of biochemical processes driven by the interaction among reactions in living cells. A reaction system consists of a set of objects $S$ representing molecules and a set of rewrite rules (called reactions) representing chemical reactions. A reaction is a triple $(R, I, P)$, where $R$, $I$ and $P$ are sets of objects representing reactants, inhibitors and products, respectively. A state of a reaction system is a subset of $S$ representing available molecules. The presence of an object in the state expresses the fact that the corresponding biological entity, in the real system being modeled, is present in a number of copies as high as needed. This is called the threshold supply assumption and characterizes reaction systems as a qualitative modeling formalism.

The dynamics of a reaction system is given by occurrences of reactions (i.e. rewrite rule applications) based on two opposite mechanisms: facilitation and inhibition. Facilitation means that a reaction can occur only if all its reactants are present, while inhibition means that the reaction cannot occur if any of its inhibitors is present. The threshold supply assumption ensures that different reactions never compete for their reactants, and hence all the applicable reactions in a step are always applied. The application of a set of reactions results in the introduction of all of their products in the next state of the system. Reaction systems assume the non permanency of the elements,
namely the next state consists only of the products of the reactions applied in the current step.

The overall behavior of a reaction system model is driven by the (set of) contextual elements which are received from the external environment at each step. Such elements join the current state of the system and, as the other objects in the system state, can enable or disable reactions. The computation of the next state of a reaction system is a deterministic procedure. However, since the contextual elements that can be received at each step can be any subset of the support set \( S \), the overall system dynamics is non-deterministic.

Theoretical aspects of reaction systems have been studied in \([7, 3, 8, 4, 9, 13, 12]\). In \([3]\) Brijder, Ehrenfeucht and Rozemberg introduced the idea of predictor. Assume that one is interested in knowing whether an object \( s \in S \) will be present after \( n \) steps of execution of a reaction system. Since the only source of non-determinism are the contextual elements received at each step, observing such elements can allow us to predict the production of \( s \) after \( n \) steps. In general, not all contextual elements are relevant for determining if \( s \) will be produced. A predictor is hence the subset \( Q \) of \( S \) that is actually essential to be observed among contextual elements for predicting whether \( s \) will be produced after \( n \) steps or not.

In \([1]\) we continued the investigation on predictors by introducing the new notion of formula based predictor. Formula based predictors consist in a propositional logic formula to be satisfied by the sequence of (sets of) elements provided by the environment. Satisfaction of the logic formula precisely discriminates the cases in which \( s \) will be produced after \( n \) steps from those in which it will not.

Formula based predictors (as well as standard predictors) do not assume anything about the elements provided by the environment. However, it is very often the case that the sequences of sets of object provided by the environment follow specific patterns or, more generally, have specific dynamical properties. For example, it might happen that some object are never provided by the environment, or that some objects are provided only after some others.

In this paper we revise formula based predictors by introducing specialized formula based predictors. The revised notion is specialized with respect to a temporal logic formula expressing the dynamical properties of the environment. More specifically, a specialized formula based predictor is a propositional logical formula that predicts the production of an object \( s \) after \( n \) steps, by considering only the subset of the context sequences satisfying the given temporal logic formula. A specialized predictor can be substantially a simpler formula than the corresponding (non-specialized) formula based predictor.

We illustrate specialized formula based predictors on a model of the lac operon expression in the \( E. coli \) bacterium originally proposed in \([5]\).

## 2 Reaction Systems

In this section we recall the basic definition of reaction systems \([10, 2]\). Let \( S \) be a finite set of symbols, called objects. A reaction is formally a triple \((R, I, P)\) with \( R, I, P \subseteq S \), composed of reactants \( R \), inhibitors \( I \), and products \( P \). We assume reactants and
inhibitors to be disjoint ($R \cap I = \emptyset$), otherwise the reaction would never be applicable. Reactants and inhibitors $R \cup I$ of a reaction are collectively called resources of such a reaction. The set of all possible reactions over a set $S$ is denoted by $\text{rac}(S)$. Finally, a reaction system is a pair $A = (S,A)$, with $S$ being the finite background set, and $A \subseteq \text{rac}(S)$ being its set of reactions.

The state of a reaction system is described by a set of objects. Let $a = (R_a, I_a, P_a)$ be a reaction and $T$ a set of objects. The result $\text{res}_a(T)$ of the application of $a$ to $T$ is either $P_a$, if $T$ separates $R_a$ from $I_a$ (i.e. $R_a \subseteq T$ and $I_a \cap T = \emptyset$), or the empty set $\emptyset$ otherwise. The application of multiple reactions at the same time occurs without any competition for the used reactants (threshold supply assumption). Therefore, each reaction which is not inhibited can be applied, and the result of application of multiple reactions is cumulative. Formally, given a reaction system $A = (S,A)$, the result of application of $A$ to a set $T \subseteq S$ is defined as $\text{res}_A(T) = \bigcup_{a \in A} \text{res}_a(T)$.

The dynamics of a reaction system is driven by the contextual objects, namely the objects which are supplied to the system by the external environment at each step. An important characteristic of reaction systems is the assumption about the non-permanency of objects. Under such an assumption the objects carried over to the next step are only those produced by reactions. All the other objects vanish, even if they are not involved any reaction.

Formally, the dynamics of a reaction system $A = (S,A)$ is defined as an interactive process $\pi = (\gamma, \delta)$, with $\gamma$ and $\delta$ being finite sequences of sets of objects called the context sequence and the result sequence, respectively. The sequences are of the form $\gamma = C_0, C_1, \ldots, C_n$ and $\delta = D_0, D_1, \ldots, D_n$ for some $n \geq 1$, with $C_i, D_i \subseteq S$, and $D_0 = \emptyset$. Each set $D_i$, for $i \geq 1$, in the result sequence is obtained from the application of reactions $A$ to a state composed of both the results of the previous step $D_{i-1}$ and the objects $C_{i-1}$ from the context; formally $D_i = \text{res}_A(C_{i-1} \cup D_{i-1})$ for all $1 \leq i \leq n$.

Finally, the state sequence of $\pi$ is defined as the sequence $W_0, W_1, \ldots, W_n$, where $W_i = C_i \cup D_i$ for all $1 \leq i \leq n$. In the following we d’say that $\gamma = C_0, C_1, \ldots, C_n$ is a $n$-step context sequence.

### 3 Preliminaries

In order to describe the causes of a given product, we use objects of reaction systems as propositional symbols of formulas. Formally, we introduce the set $F_S$ of propositional formulas on $S$ defined in the standard way: $S \cup \{\text{true}, \text{false}\} \subseteq F_S$ and $\neg f_1, f_1 \lor f_2, f_1 \land f_2 \in F_S$ if $f_1, f_2 \in F_S$.

The propositional formulas $F_S$ are interpreted with respect to subsets of the objects $C \subseteq S$. Intuitively, $s \in C$ denotes the presence of element $s$ and therefore the truth of the corresponding propositional symbol. The complete definition of the satisfaction is as follows.

**Definition 1.** Let $C \subseteq S$ for a set of objects $S$. Given a propositional formula $f \in F_S$, the satisfaction relation $C \models f$ is inductively defined as follows:

- $C \models s$ iff $s \in C$,
- $C \models \neg f'$ iff $C \not\models f'$,
- $C \models f_1 \land f_2$ iff $C \models f_1$ and $C \models f_2$.

- $C \models f_1 \lor f_2$ iff either $C \models f_1$ or $C \models f_2$,
In the following \( \equiv_i \) stands for the logical equivalence on propositional formulas \( F_S \). Moreover, given a formula \( f \in F_S \) we use \( \text{atom}(f) \) to denote the set of propositional symbols that appear in \( f \) and \( \text{simpl}(f) \) to denote the simplified version of \( f \). The simplified version of a formula is obtained applying the standard formula simplification procedure of propositional logic converting a formula to Conjunctive Normal Form. We recall that for any formula \( f \in F_S \) the simplified formula \( \text{simpl}(f) \) is equivalent to \( f \), it is minimal with respect to the propositional symbols. Thus, we have \( f \equiv_i \text{simpl}(f) \) and \( \text{atom}(\text{simpl}(f)) \subseteq \text{atom}(f) \) and there exists no formula \( f' \) such that \( f' \equiv_i f \) and \( \text{atom}(f') \subset \text{atom}(\text{simpl}(f)) \).

The causes of an object in a reaction system are defined by a propositional formula on the set of objects \( S \). First of all we define the applicability predicate of a reaction \( a \) as a propositional logic formula on \( S \) describing the requirements for applicability of \( a \), namely that all reactants have to be present and inhibitors have to be absent. This is represented by the conjunction of all atomic formulas representing reactants and the negations of all atomic formulas representing inhibitors of the considered reaction.

**Definition 2.** Let \( a = (R,I,P) \) be a reaction with \( R,I,P \subseteq S \) for a set of objects \( S \). The applicability predicate of \( a \), denoted by \( \text{ap}(a) \), is defined as follows:
\[
\text{ap}(a) = (\bigwedge_{s_i \in R} s_i) \land (\bigwedge_{s_i \in I} \neg s_i).
\]

The causal predicate of a given object \( s \) is a propositional formula on \( S \) representing the conditions for the production of \( s \) in one step, namely that at least one reaction having \( s \) as a product has to be applicable.

**Definition 3.** Let \( A = (S,A) \) be a r.s. and \( s \in S \). The causal predicate of \( s \) in \( A \), denoted by \( \text{cause}(s,A) \) (or \( \text{cause}(s) \)), when \( A \) is clear from the context, is defined as follows:
\[
\text{cause}(s,A) = \bigvee_{(R,I,P) \in A} \text{ap}((R,I,P)).
\]

We introduce a simple reaction system as running example.

**Example 1.** Let \( A = (\{A,\ldots,G\},\{a_1,a_2,a_3\}) \) be a reaction system with
\[
a_1 = (\{A\},\{\}\},\{B\}) \quad a_2 = (\{C,D\},\{\},\{E,F\}) \quad a_3 = (\{G\},\{B\},\{E\}).
\]

The applicability predicates of the reactions are \( \text{ap}(a_1) = A \), \( \text{ap}(a_2) = C \land D \) and \( \text{ap}(a_3) = G \land \neg B \). Thus, the causal predicates of the objects are
\[
\text{cause}(A) = \text{cause}(C) = \text{cause}(D) = \text{cause}(G) = \text{false},
\]
\[
\text{cause}(B) = A, \quad \text{cause}(F) = C \land D, \quad \text{cause}(E) = (G \land \neg B) \lor (C \land D)
\]

Note that \( \text{cause}(A) = \text{false} \) given that \( A \) cannot be produced by any reaction. An analogous reasoning holds for objects \( C, D \) and \( G \).

### 4 Formula Based Predictors

We introduce the notion of formula based predictors, originally presented in [1]. A formula based predictor for an object \( s \) at step \( n+1 \) is a propositional formula satisfied

\footnote{We assume that \( \text{cause}(s) = \text{false} \) if there is no \( (R,I,P) \in A \) such that \( s \in P \).}
exactly by the context sequences leading to the production of \( s \) at step \( n + 1 \). Minimal formula based predictors can be calculated in an effective way.

Given a set of objects \( S \), we consider a corresponding set of *labelled objects* \( S \times \mathbb{N} \). For the sake of legibility, we denote \((s, i) \in S \times \mathbb{N}\) simply as \( s_i \) and we introduce \( S^n = \bigcup_{i=0}^{\infty} S_i \) where \( S_i = \{ s_i \mid s \in S \} \). Propositional formulas on labelled objects \( S^n \) describe properties of \( n \)-step context sequences. The set of propositional formulas on \( S^n \), denoted by \( F_{S^n} \), is defined analogously to the set \( F_S \) (presented in Sect. 3) by replacing \( S \) with \( S^n \). The satisfaction relation of Def. 1 applies also to formulas in \( F_{S^n} \) on subsets of \( S^n \).

A labelled object \( s_i \) represents the presence (or the absence, if negated) of object \( s \) in the \( i \)-th element \( C_i \) of the \( n \)-step context sequence \( \gamma = C_0, C_1, \ldots, C_n \). This interpretation leads to the following definition of satisfaction relation for propositional formulas on context sequences.

**Definition 4.** Let \( \gamma = C_0, C_1, \ldots, C_n \) be a \( n \)-step context sequence and \( f \in F_{S^n} \) a propositional formula. The satisfaction relation \( \gamma \models f \) is defined as

\[
\{ s_i \mid s \in C_i, 0 \leq i \leq n \} \models f.
\]

As an example, let us consider the context sequence \( \gamma = C_0, C_1 \) where \( C_0 = \{ A, C \} \) and \( C_1 = \{ B \} \). We have that \( \gamma \) satisfies the formula \( A_0 \land B_1 \) (i.e. \( \gamma \models A_0 \land B_1 \)) while \( \gamma \) does not satisfy the formula \( A_0 \land (\neg B_1 \lor C_1) \) (i.e. \( \gamma \not\models A_0 \land (\neg B_1 \lor C_1) \)).

The latter notion of satisfaction allows us to define formula based predictor.

**Definition 5 (Formula based Predictor).** Let \( \mathcal{A} = (S, A) \) be a r.s. \( s \in S \) and \( f \in F_{S^n} \) a propositional formula. We say that \( f \) \( f \)-predicts \( s \) in \( n + 1 \) steps if for any \( n \)-step context sequence \( \gamma = C_0, \ldots, C_n \)

\[
\gamma \models f \iff s \in D_{n+1}
\]

where \( \delta = D_0, \ldots, D_n \) is the result sequence corresponding to \( \gamma \) and \( D_{n+1} = \text{res}_{\mathcal{A}}(C_n \cup D_n) \).

Note that if formula \( f \) \( f \)-predicts \( s \) in \( n + 1 \) steps and if \( f' \equiv f \) then also \( f' \) \( f \)-predicts \( s \) in \( n + 1 \). More specifically, we are interested in the formulas that \( f \)-predict \( s \) in \( n + 1 \) and contain the minimal numbers of propositional symbols, so that their satisfiability can easily be verified. This is formalised by the following approximation order on \( F_{S^n} \).

**Definition 6 (Approximation Order).** Given \( f_1, f_2 \in F_{S^n} \) we say that \( f_1 \sqsubseteq_f f_2 \) if and only if \( f_1 \equiv_f f_2 \) and \( \text{atom}(f_1) \subseteq \text{atom}(f_2) \).

It can be shown that there exists a *unique equivalence class* of formulas based predictors for \( s \) in \( n + 1 \) steps that is minimal w.r.t. the order \( \sqsubseteq_f \).

We now define an operator \( \text{fbp} \) that allows formula based predictors to be effectively computed.
Definition 7. Let \( A = (S,A) \) be a r.s. and \( s \in S \). We define a function \( \text{fbp} : S \times \mathbb{N} \rightarrow F_S \) as follows: \( \text{fbp}(s,n) = \text{fbs}(\text{cause}(s),n) \), where the auxiliary function \( \text{fbs} : F_S \times \mathbb{N} \rightarrow F_S \) is recursively defined as follows:

\[
\begin{align*}
\text{fbs}(s,0) &= s_0 \\
\text{fbs}(s,i) &= s_i \cup \text{fbs}(\text{cause}(s),i-1) \text{ if } i > 0 \\
\text{fbs}((f'),i) &= (\text{fbs}(f',i)) \\
\text{fbs}(\neg f',i) &= \neg \text{fbs}(f',i) \\
\text{fbs}(\text{true},i) &= \text{true} \\
\text{fbs}(\text{false},i) &= \text{false}
\end{align*}
\]

The function \( \text{fbp} \) gives a formula based predictor that, in general, may not be minimal w.r.t. to \( \sqsubseteq_f \). Therefore, the calculation of a minimal formula based predictor requires the application of a standard simplification procedure to the obtained logic formula.

Theorem 1. Let \( A = (S,A) \) be a r.s.. For any object \( s \in S \),

- \( \text{fbp}(s,n) \) \( f \)-predicts \( s \) in \( n + 1 \) steps;
- \( \text{simpl}(\text{fbp}(s,n)) \) \( f \)-predicts \( s \) in \( n + 1 \) steps and is minimal w.r.t. \( \sqsubseteq_f \).

Example 2. Let us consider again the reaction system of Ex. 1. We are interested in the production of \( E \) after 4 steps. Hence, we calculate the logic formula that \( f \)-predicts \( E \) in 4 steps applying the function \( \text{fbp} \):

\[
\text{fbp}(E,3) = \text{fbs}((G \land \neg B) \lor (C \land D),3) \\
= (\text{fbs}(G,3) \land \neg \text{fbs}(B,3)) \lor (\text{fbs}(C,3) \land \text{fbs}(D,3)) \\
= ((G_3) \land \neg (B_3 \lor \text{fbs}(A,2))) \lor (C_3 \land D_3) \\
= (G_3 \land \neg B_3 \land \neg A_2) \lor (C_3 \land D_3)
\]

A context sequence satisfies \( \text{fbp}(E,3) \) iff the execution of the reaction system leads to the production of object \( E \) after 4 steps. Furthermore, in this case the obtained formula is also minimal given that \( \text{simpl}(\text{fbp}(E,3)) = \text{fbp}(E,3) \).

5 The Temporal Logic for Context Sequences

We introduce a linear temporal logic for the description of properties of finite context sequences. In the logic, propositional formulas describe the properties of single contexts (i.e. the symbols that can/cannot appear in an element of a context sequence). Hence, such formulas play the role of state formulas in traditional temporal logics. Temporal properties are expressed by variants of the usual \( \text{next} \) and \( \text{until} \) operators, and by derived \( \text{eventually} \) and \( \text{globally} \) operators.

Definition 8 (Temporal Formulas). Let \( S \) be a set of objects. The syntax of temporal logic formulas on \( S \) is defined by the following grammar:

\[
\psi ::= f \mid \psi \lor \psi \mid \psi \land \psi \mid X\psi \mid \psi U_k \psi \mid F_k \psi \mid G_k \psi
\]

where \( f \in F_s \) and \( k \in \mathbb{N} \cup \{\infty\} \). We denote with \( TL_S \) the set of all temporal logic formulas on \( S \).
Given a context sequence \( \gamma = C_0, \ldots, C_n \), the simple temporal formula \( f \) states that \( f \) is satisfied by \( C_0 \), according to the definition of satisfiability of the propositional logic. Formula \( X \psi \) states that \( \psi \) is satisfied by the context sequence after one step, namely by \( C_1, \ldots, C_n \). Formula \( \psi_1 \cup \psi_2 \) states that there exists \( k' \leq k \) such that \( C_0, \ldots, C_{k'} \) satisfies \( \psi_1 \) and \( C_{k'} \) satisfies \( \psi_2 \). Formula \( F_k \psi \) states that there exists \( k' \leq k \) such that \( C_{k'} \) satisfies \( \psi \). Formula \( G_k \psi \) states that for all \( i \leq k \) context \( C_i \) satisfies \( \psi \). Finally, operators \( \lor \) and \( \land \) are as usual.

Note that the value of \( n \) has a significant role in the satisfiability of temporal formulas. In particular, if \( n < k \) we have that \( \psi_1 \cup \psi_2 \) is satisfied also if \( \psi_1 \) is satisfied by all \( C_i \) with \( i \leq n \) (never satisfying \( \psi_2 \)). Moreover, if \( n < k \) we have that \( F_k \psi \) is always satisfied since, intuitively, \( \psi \) may be satisfied by a context \( C_j \), with \( n < j \leq k \), that is not contained in \( \gamma \). An analogous reasoning holds for \( X \psi \) when \( n = 0 \) (or \( XX \psi \) when \( n = 1 \), etc.). As regards \( G_k \psi \), when \( n < k \) it is equivalent to \( G_n \psi \).

As usual in temporal logic, formulas \( F_k \psi \) and \( G_k \psi \) are actually syntactic sugar for \( true \cup \psi \) and \( \psi \land false \), respectively. Moreover, if \( k \in \mathbb{N} \), we have that also \( \psi_1 \cup_k \psi_2 \) can be rewritten into \( \psi_2 \lor (\psi_1 \land X(\psi_1 \cup_{k-1} \psi_2)) \), when \( k > 0 \), and \( \psi_2 \), when \( k = 0 \). Consequently, in the semantics of the temporal logic, we can omit derived operators \( F_k \) and \( G_k \), with \( k \in \mathbb{N} \cup \{ \infty \} \), and \( U_k \) with \( k \in \mathbb{N} \).

The formal definition of satisfiability of temporal logic formulas on finite \( n \)-step context sequences is as follows.

**Definition 9.** Let \( \gamma = C_0, C_1, \ldots, C_n \) be a \( n \)-step context sequence with \( n \geq 0 \). Given a temporal logic formula \( \psi \in TL_S \), the satisfaction relation \( \gamma \vdash \psi \) is inductively defined as follows:

\[
\gamma \vdash f \iff C_0 \models f \\
\gamma \vdash \psi_1 \land \psi_2 \iff \gamma \vdash \psi_1 \land \gamma \vdash \psi_2 \\
\gamma \vdash \psi_1 \lor \psi_2 \iff \text{either } \gamma \vdash \psi_1 \text{ or } \gamma \vdash \psi_2 \\
\gamma \vdash \psi_1 \cup \psi_2 \iff \text{either } \gamma \vdash \psi_1 \\
\text{or } \gamma \vdash \psi_1 \text{ and if } n > 0 \text{ then } \gamma \vdash \psi_1 \cup \psi_2
\]

where \( \gamma' = C_0, \ldots, C_{n-1} \) with \( C_i' = C_{i+1} \) for \( 0 \leq i \leq n-1 \).

Finally, we present an encoding of temporal formulas on objects \( S \) into propositional formulas on labelled objects \( S^n \). The encoding depends on the parameter \( n \in \mathbb{N} \) reporting the length of the context sequences that we want to model.

**Definition 10.** Let \( S \) be a set of objects and \( n \in \mathbb{N} \). The encoding of a temporal logic formula \( \psi \in TL_S \) into a propositional logic formula on \( S^n \) is given by \( [\psi]_n \) where the function \( [ ] : TL_S \times \mathbb{N} \rightarrow F_{S^n} \) is defined as follows:

\[
[f]_n = [f]_i \\
[X\psi]_n = \begin{cases} 
[\psi]_{n+1} & \text{if } i < n \\
true & \text{if } i = n
\end{cases} \\
[\psi_1 \lor \psi_2]_n = [\psi_1]_n \lor [\psi_2]_n \\
[\psi_1 \land \psi_2]_n = [\psi_1]_n \land [\psi_2]_n \\
[\psi_1 \cup \psi_2]_n = \begin{cases} 
[\psi_2]_n \lor ( [\psi_1]_n \land [\psi_1 \cup \psi_2]_{i+1} ) & \text{if } i < n \\
[\psi_2]_n \lor [\psi_1]_n & \text{if } i = n
\end{cases}
\]
where \( \lfloor \cdot \rfloor_i : F_S \to F_S \) is a function that replaces, in a given propositional logic formula \( f \), every \( s \in S \) with the corresponding labelled object \( s_i \in S_i \).

The following theorem states the main property of the encoding of temporal formulas: the result of the encoding of a temporal formula is equivalent.

**Theorem 2.** Let \( \gamma = C_0, \ldots, C_n \) be a \( n \)-step context sequence. For any temporal formula \( \psi \in TL_S \) it holds that \( \gamma \vdash \psi \iff \gamma \models [\psi]_0^n \).

### 6 Specialized Formula Based Predictors

We specialize the notion of formula based predictor (given in Def. 5) w.r.t. a subset of context sequences characterized by a temporal logic formula \( \psi \).

**Definition 11 (Specialized Formula based Predictor).** Let \( A = (S, A) \) be a r.s., \( s \in S \), \( f \in F_S^n \) and \( \psi \in TL_S \). We say that \( f \) \( f \)-predicts \( s \) in \( n + 1 \) steps with respect to \( \psi \) iff for any \( n \)-step context sequence \( \gamma = C_0, \ldots, C_n \) such that \( \gamma \vdash \psi \) we have that

\[
\gamma \models f \iff s \in D_{n+1}
\]

where \( \delta = D_0, \ldots, D_n \) is the result sequence corresponding to \( \gamma \) and \( D_{n+1} = \text{res}_A(C_n \cup D_n) \).

It should be clear that any formula \( f \) that \( f \)-predicts \( s \) in \( n + 1 \) steps also \( f \)-predicts \( s \) in \( n + 1 \) steps with respect to any logical formula \( \psi \). In particular, the formula \( fbp(s, n) \) (and its simplified version \( \text{simpl}(fbp(s, n)) \)) is a specialized predictor for \( s \) in \( n + 1 \) steps for any \( \psi \). However, the formula \( \text{simpl}(fbp(s, n)) \) typically is too general and therefore is not a minimal (w.r.t. \( \subseteq_f \)) formula based predictor specialized w.r.t. \( \psi \).

Thus, we introduce a methodology that allows us to calculate minimal formula based predictors for an object \( s \) at \( n + 1 \) step, specialized w.r.t. \( \psi \). The idea is to use the encoding of the temporal formula \( \psi \) computed with respect to the length of the context sequences \( n \). The encoding \( [\psi]_0^n \) is a propositional formula on labelled objects \( S^n \) that models the \( n \)-step context sequences satisfying \( \psi \). A formula based predictor specialized w.r.t. \( \psi \) can be derived by exploiting the encoding \( [\psi]_0^n \) to simplify a corresponding formula based predictor.

More formally, suppose that \( f \) \( f \)-predicts \( s \) in \( n + 1 \) steps. A formula \( f' \) \( f \)-predicts \( s \) in \( n + 1 \) steps w.r.t. a temporal formula \( \psi \) whenever \( f' \) is logically equivalent to \( f \) considering the context sequences satisfying \( \psi \), that is assuming that the conditions formalized by the formula \( [\psi]_0^n \) holds. The following theorem establishes this fundamental property of specialized formula based predictors.

**Theorem 3.** Let \( A = (S, A) \) be a r.s., \( s \in S \), and \( f \in F_S^n \) such that \( f \) \( f \)-predicts \( s \) in \( n + 1 \) steps. Given \( \psi \in TL_S \) and \( f' \in F_S^n \) such that

\[
[\psi]_0^n \Rightarrow (f \equiv_f f')
\]

we have that \( f' \) \( f \)-predicts \( s \) in \( n + 1 \) steps with respect to \( \psi \).
We introduce a minimization algorithm that allows us to calculate a minimal formula \( f' \) that satisfies the property of Theorem 3, given a predictor \( f \) and a temporal logic formula \( \psi \).

Let \( B = \{ \text{true, false} \} \) and consider the incompletedly specified boolean function \( F: B^n \rightarrow B \) composed of the following two disjoint subsets \( \text{Yes}, \text{DontC} \) of \( B^n \).

\[
\text{Yes}_F = \{ x_1, \ldots, x_n \mid \text{\text{true}} \}, \quad \text{DontC}_F = \{ x_1, \ldots, x_n \mid \text{\text{false}} \}
\]

Function \( F \) is evaluated to \( \text{true} \) if \( \text{Yes}_F \) and \( \text{DontC}_F \) are both \( \text{true} \), otherwise \( F \) is evaluated to \( \text{false} \). Each \( F \) has the minimal number of variables.

Consider any possible extension of \( F \), let us call them \( F_1, \ldots, F_m \), obtained by considering \( \text{Yes}_F = \text{Yes}_F \cup D_i \) with \( D_i \subseteq \text{DontC}_F \) and \( \text{DontC}_F, = \emptyset \). Each \( F_i \) is now a completely specified boolean function and it can be shown that in this case its minimal sum of product form (SOP) also contains a minimal number of different variables. Hence, any algorithm that computes the minimal SOP form of \( F \) (i.e. any well known algorithm looking for the prime implicants of \( F \)) can be applied in order to minimize the number of variables of \( F \). Then consider the \( F' \) corresponding to the minimal SOP form between all the minimal SOP form of \( F_1, \ldots, F_m \), we are guaranteed that such \( F' \) has the minimal number of variables.

It is worth noting that considering all possible extensions of the incompletedly specified boolean function \( F \) as well computing the SOP form of a completely specified boolean function \( F^0 \) has an exponential cost. However, some heuristic that, in general, cannot guarantee minimality can be designed in order to improve the efficiency.

**Example 3.** Let us consider the r.s. of Ex. 1 with the following reactions:

\[
a_1 = \{ \{A\}, \emptyset, \{B\} \} \quad a_2 = \{ \{C,D\}, \emptyset, \{E,F\} \} \quad a_3 = \{ \{G\}, \{B\}, \{E\} \}.
\]

We introduce the following temporal formulas in order to describe the properties of the context sequences:

\[
\psi_1 = G_{\infty}(\neg B \land (\neg C \lor D)), \quad \psi_2 = G_{\infty}B, \quad \psi_3 = F_3B
\]

Formula \( \psi_1 \) describes an invariant property which holds at any step of the context sequence. The property models the situation in which object \( B \) is not a synthesizable product, i.e., it can not be found in the environment but it has to be produced by the reactions. Moreover, if the environment supplies \( C \) it also supplies \( D \) at the same time. Similarly, formula \( \psi_2 \) shows that the environment always supply the object \( B \). By contrast, formula \( \psi_3 \) says that the environment will eventually supply the object \( B \) within 3 steps.

We show the specialized formula based predictors w.r.t. the previous temporal formulas considering again the production of \( E \) in 4-steps. We recall that the corresponding minimal formula based predictor (presented in Ex. 2) is

\[
simpl(fbp(E, 3)) = fbp(E, 3) = (G_3 \land \neg B_3 \land \neg A_2) \lor (C_3 \land D_3).
\]
In order to calculate the specialized versions of predictor we compute the encoding of the temporal formulas obtaining:

\[ [\psi_1]^3 = (\neg B_0 \lor \neg B_1 \lor \neg B_2 \lor \neg B_3) \land (\neg C_0 \lor D_0) \land (\neg C_1 \lor D_1) \land (\neg C_2 \lor D_2) \land (\neg C_3 \lor D_3), \]

\[ [\psi_2]^3 = B_0 \land B_1 \land B_2 \land B_3, \quad [\psi_3]^3 = B_0 \lor B_1 \lor B_2 \lor B_3. \]

By applying the minimization algorithm to \( \text{fbp}(E, 3) \) and to \( \psi_1, \psi_2 \) and \( \psi_3 \), we obtain the corresponding specialized formulas \( f_1, f_2 \) and \( f_3 \), respectively:

\[ f_1 = (G_3 \land \neg A_2) \lor C_3, \quad f_2 = (C_3 \land D_3), \quad f_3 = \text{fbp}(E, 3). \]

In the case of formulas \( \psi_1 \) and \( \psi_2 \), the specialized formulas \( f_1 \) and \( f_2 \) are substantially reduced w.r.t. the formula based predictor, while formula \( \psi_3 \) does not lead to a reduced specialized predictor.

7 Application

In this section we introduce a more complex biological example: the lac operon expression in the \( E. coli \) bacterium. The lactose operon is a sequence of genes that are responsible for producing enzymes for lactose degradation. We borrow the formalization of this biological system as a reaction system from [5]. Let \( \mathcal{A} = (S, \{a_1, \ldots, a_{10}\}) \) be the reaction system where

\( S = \{\text{lac, lacI, I, I-OP, cya, cAMP, crp, CAP, cAMP-CAP, lactose, glucose, Z, Y, A}\} \)

and the reaction rules are defined as follows

\[ a_1 = (\{\text{lac}\}, \{\}\), \{lac\}) \quad \text{(lac operon duplication)} \]
\[ a_2 = (\{\text{lacI}\}, \{\}\), \{lacI\}) \quad \text{(repressor gene duplication)} \]
\[ a_3 = (\{\text{lacI}\}, \{\}\), \{I\}) \quad \text{(repressor gene expression)} \]
\[ a_4 = (\{I\}, \{\text{lactose}\}, \{I-OP\}) \quad \text{(regulation mediated by lactose)} \]
\[ a_5 = (\{\text{cyA}\}, \{\}\), \{cyA\}) \quad \text{(cyA duplication)} \]
\[ a_6 = (\{\text{cyA}\}, \{\}\), \{cAMP\}) \quad \text{(cAMP expression)} \]
\[ a_7 = (\{\text{crp}\}, \{\}\), \{crp\}) \quad \text{(crp duplication)} \]
\[ a_8 = (\{\text{crp}\}, \{\}\), \{CAP\}) \quad \text{(crp expression)} \]
\[ a_9 = (\{\text{cAMP, CAP}\}, \{\text{glucose}\}, \{\text{cAMP-CAP}\}) \quad \text{(regulation mediated by glucose)} \]
\[ a_{10} = (\{\text{lac, cAMP-CAP}\}, \{\text{I-OP}\}, \{Z, Y, A\}) \quad \text{(lac operon expression)} \]

The regulation process is as follows: gene \( \text{LacI} \) encodes the lac repressor \( I \), which, in the absence of lactose, binds to gene \( \text{OP} \) (the operator). Transcription of structural genes into mRNA is performed by the RNA polymerase enzyme which transcribes the three structural genes represented by \( \text{lac} \) into a single mRNA fragment. When the lac repressor \( I \) is bound to gene \( \text{OP} \) (that is, the complex \( I-\text{OP} \) is present) it becomes an obstacle for the RNA polymerase, and transcription of the structural genes is not performed. On the other hand, when lactose is present inside the bacterium, it binds to the repressor thus inhibiting the binding of \( I \) to \( \text{OP} \). This inhibition allows the transcription of genes represented by \( \text{lac} \) by the RNA polymerase.
Two more genes encode for the production of two particular proteins: \textit{cAMP} and \textit{CAP}. These genes are called, respectively, \textit{cya} and \textit{crp}, and they are indirectly involved in the regulation of the lac operon expression. When glucose is not present, \textit{cAMP} and \textit{CAP} proteins can produce the complex \textit{cAMP-CAP} which can increase significantly the expression of \textit{lac} genes. Also in presence of the \textit{cAMP-CAP} complex, the expression of the \textit{lac} genes is inhibited by \textit{I-OP}.

Note that the reactions \{a_1, a_2, a_5, a_7\} are needed to ensure the permanency of the genes in the system.

In [5] the authors investigate the effects on the production of enzymes \textit{Z}, \textit{Y} and \textit{A} when the environment provides both glucose and lactose, only glucose, only lactose, or none of them. The genomic elements \textit{lac}, \textit{lacI}, \textit{cya} and \textit{crp} together with the proteins \textit{I}, \textit{cAMP} and \textit{CAP}, that are normally present in the bacterium, are supplied to the system by the starting context \textit{C_0}. Then, an example context sequence \(\gamma = C_0, \ldots, C_{40}\) is considered, in which every element \(C_i\) with \(1 < i <= 40\) is a subset of \{\textit{glucose}, \textit{lactose}\}. Such a context sequence represents an environment in which the supply of glucose and lactose varies over time. By observing the result states \(D_1, \ldots, D_{40}\) obtained by executing the reaction system, the authors conclude that the enzymes \textit{Z}, \textit{Y} and \textit{A} are produced in a step \(i\) only if \textit{lactose} was the only element provided to the system two steps before. Formally, this can be expressed as follows:

\[
Z, Y, A \in D_i \text{ iff } C_{i-2} = \{\textit{lactose}\}, \text{ with } i > 3.
\]

This conclusion has been reached empirically, by observing a single execution of the system with respect to an example context sequence. The conditions for the production of \textit{Z}, \textit{Y} and \textit{A} can instead be studied by applying the notions of predictor we defined in this paper.

For sake of simplicity we consider the formula based predictors for the enzymes \textit{Z}, \textit{Y} and \textit{A} in 4 steps, noting that the effects of all reactions can be observed after four steps. We obtain the following formula

\[
\text{fbp}(Z, 3) = \text{fbp}(Y, 3) = \text{fbp}(A, 3) = \\
((\text{lac}_3 \lor \text{lac}_2 \lor \text{lac}_1 \lor \text{lac}_0) \land (\text{cAMP} \land \text{CAP}_3) \lor ((\text{cAMP}_2 \lor \text{cya}_1 \lor \text{cy}_a_0) \land (\text{CAP}_2 \land \text{crp}_1 \lor \text{crp}_0) \land \neg \text{glucose}_2)) \\
\land ((\neg \text{IOP}_3 \land \neg \text{I}_2 \land \neg \text{lac}_{I_1} \land \neg \text{lac}_{I_0} \lor \text{lactose}_2)).
\]

The obtained formula is minimal w.r.t. \(\sqsubseteq_f\) and therefore does not require the application of simplification techniques. Since we are interested in studying context sequences that supply only \textit{glucose} and \textit{lactose} it is useful to specialize the previous formula based predictor for context sequences described by the following temporal formula:

\[
\psi = f_{\text{initial}} \land G_{\infty} \neg f_{\text{internal}} \land XG_{\infty} \neg f'_{\text{initial}}
\]

where \(f_{\text{initial}} = \text{lac} \land \text{lacI} \land \text{cy}_a \land \text{crp} \land I \land \text{cAMP} \land \text{CAP}, f'_{\text{initial}} = \text{lac} \land \text{lacI} \lor \text{cy}_a \lor \text{crp} \lor I \lor \text{cAMP} \lor \text{CAP}\) and \(f_{\text{internal}} = I \lor \text{IOP} \lor \text{cAMP} \lor \text{CAP} \lor Z \lor Y \lor A\).

Let us focus on the production of \textit{A}, noting that, since \(\text{fbp}(Z, 3) = \text{fbp}(Y, 3) = \text{fbp}(A, 3)\), the same reasoning apply to enzymes \textit{Z} and \textit{Y}. By applying the minimization procedure described in Sect. 6 to \(\text{fbp}(A, 3)\) and \(\psi\), we obtain the following reduced
predictor of $A$ in 4 steps specialized w.r.t. $\psi$

$$f' = \neg \text{glucose}_2 \land \text{lactose}_2.$$  

Formula $f'$ clearly shows that the enzymes $A$ is produced at step 4 only if lactose was actually the only element provided to the system two steps before.

The specialized predictor $f'$ we obtained agrees with the conclusion reached in [5] by reasoning on an example of context sequence.

Note that the same conclusion was reached in [1] by simplifying predictor $\text{fbp}(A, 3)$ on the basis of informal reasoning on the properties of context sequences. In this paper, we have been able to obtain $f'$ in a fully formalized and automatic way, on the basis of $\text{fbp}(A, 3)$ and $\psi$.

8 Conclusions

We have presented a revised notion of formula based predictor [1] in which the predictor is specialized with respect to a temporal logic formula $\psi$ expressing the properties of the context sequences. More specifically, the formula based predictor models the necessary conditions for an object $s$ to be produced in $n$ steps, assuming that the reaction system is executed with respect to a context sequence satisfying $\psi$. The advantage of the specialized version of predictor is that the resulting propositional formula can be substantially reduced with respect to the corresponding formula based predictor. As a consequence, this approach is very convenient whenever we are interested to observe whether an element $s$ will be produced after $n$ steps or not for a certain class of context sequences rather than for any possible context sequences.

As future work we plan to consider the application of the tabling and generalization procedures described in [1] to specialized predictors.

References

On Decidability of Persistence Notions

Kamila Barylska and Lukasz Mikulski
Faculty of Mathematics and Computer Science
Nicolaus Copernicus University
Torun, Chopina 12/18, Poland
{kamila.barylska,lukasz.mikulski}@mat.umk.pl

Abstract. Persistence is a widely investigated fundamental property of concurrent systems, which was extended in many ways. We propose a unified characterisation of several notions considered in the literature. The main result of the paper is a detailed description of a general and extendable framework that allows to state decision problems for different persistence notions (well known as well as newly formulated) and prove their decidability.

Key words: persistence, nonviolence, step semantics, Petri nets

1 Introduction

Conflicts, in other word, situations where two components fight for resources causing mutual exclusion, are usually not desirable in concurrent systems. In the case of Petri Nets, one can exclude them on a design level by modelling systems without statical conflicts (which are certain templates in a net structure). Examples of such classes of nets are Marked Graphs, T-systems or Output-Nonbranching Nets (ON-Nets).

The next step to avoid conflicts is to study concurrent systems of arbitrary structure which work in a conflict-free manner. Such systems are called persistent, and are widely investigated [4, 6, 14, 15, 19, 21, 24, 25] due to their applications in concurrent systems, for example in net synthesis [8], or in hardware designing [9, 23] (one can use persistent nets to avoid hazards [12]).

In its standard form, persistence is stated as a property of nets operating according to the sequential semantics. The classical definition is as follows:

Definition 1 (persistent net, [19]). A p/t-net N is persistent if, for all transitions \( a \neq b \) and any reachable marking \( M \) of \( N \), the enabledness of actions \( a \) and \( b \) at marking \( M \) implies that the sequence \( ab \) is also enabled at \( M \).

The above definition captures a property of the entire system represented by a p/t-net. We now disassemble it onto basic components. First of all, the subjects of our interest are two objects from a finite domain (e.g., transitions \( a \) and \( b \)). They should be essentially different (e.g., \( a \neq b \)) and initially enabled (e.g., \( M(a) \) and \( M(b) \)) in a specified context (e.g., a marking \( M \) reachable from \( M_0 \); note that the set of all such markings may be infinite). We test the preservation of the terminal enabledness of the second object (e.g., \( M(ab) \)). At the end, we wrap it by universal first-order logic quantifiers (over actions \( a, b \in T \) and a reachable marking \( M \in [M_0] \)).

We provide the following framework with parameters (degrees of freedom):
– the object type (steps or transitions as singleton steps)
– the existence of universal quantifiers
– the essential difference (especially in the case of steps) $\delta_{\alpha,\beta}$
– the initial enabledness of $\alpha$ (assumed to be immediate) $\zeta_{\alpha,M}$
– the initial enabledness of $\beta$ (immediate, ultimate) $\xi_{\beta,M}$
– the terminal enabledness (immediate, ultimate) $\phi_{\alpha,\beta,M}$.

Example 1. Let $N = (P, T, W, M_0)$ be a $p/t$-net.

The classical notion of persistence in $p/t$-nets with sequential semantics can be obtained with the use of the following parameters:

– the object type – transitions from $T$
– the existence of three universal quantifiers ($\forall a \in T, \forall b \in T, \forall M \in [M_0]$)
– the essential difference – $\delta_{\alpha,\beta} : a \neq b$
– the initial enabledness of $\alpha$ – $\zeta_{\alpha,M} : M[\alpha]$ 
– the initial enabledness of $\beta$ – $\xi_{\beta,M} : M[\beta]$ 
– the terminal enabledness – $\phi_{\alpha,\beta,M} : M[\alpha\beta]$.

Figure 1 illustrates a persistent $p/t$-net, in which conflicts are not excluded structurally.

Fig. 1. A persistent $p/t$-net [4].

Specifying all those parameters leads to a fixed formula, which corresponds to a distinct decision problem. Many of the problems obtained this way were discussed (sometimes without answering the question about decidability) in the literature [2, 15, 18]. However, in a natural way, some new problems appeared. The presented framework allows us to answer all of them positively.

The paper is organized as follows. First, we introduce the general framework that allows to define decision problems concerning different persistence notions and specify their decidability in terms of satisfiability of first order logic formulas. After that,
in Section 3, we describe in detail the technique of proving decidability of all problems formerly presented. Section 4 consists of several meaningful examples showing the applications of the presented framework. We end with a short section containing conclusions and future plans.

Part of the results was presented during PNSE workshop 2014 [1].

Due to lack of space the whole section of preliminaries concerning Petri Nets with step semantics is not a part of this version. It can be found in full version [3]. Basic definitions and notions could be also found e.g. in [4, 18].

2 General Framework

All (but the first) parameters of the framework presented in the introduction are used to specify the predicate (propositional logic formula) \( \psi_{\alpha,\beta,M} = \zeta_{\alpha,M} \land \xi_{\beta,M} \land \delta_{\alpha,\beta} \Rightarrow \phi_{\alpha,\beta,M} \). Having specified \( \psi \), we can focus on the decidability of universally quantified problems constructed this way. The diagram that presents the lattice of those problems is depicted in Figure 2.

![Fig. 2. Lattice of quantified formulas](image)

Each arc of the diagram corresponds to adding one universal quantifier. The thin arcs (e.g., an arc between \( \psi_{\alpha,\beta,M} \) and \( \forall M \psi_{\alpha,\beta,M} \)) are 'simple' as the set of interesting objects is assumed to be finite. Having established the decidability of a predecessor of a diagram arc, we obtain the decidability of its successor just by searching through a finite sequence of the relevant cases. Going through the solid thick arcs (e.g., an arc between \( \forall M \forall \beta \psi_{\alpha,\beta,M} \) and \( \forall M \forall \alpha \forall \beta \psi_{\alpha,\beta,M} \)) is more complex. In this case, we need to check if the predicate is true for all elements \( M \) from a (possibly infinite) subset of a (possibly infinite) set of contexts.

The general idea of processing a thick arc is as follows:

(i) negate the predicate in the head node of a diagram arc;
(ii) compute the set of all contexts in which the negated predecessor is true;
(iii) intersect this set with the set of all contexts covered by a quantifier over \( M \);
(iv) check if the resulting set is empty.
Now, we can make the following observations:

(i) The first point is easy, as a negation of a decidable predicate is obviously decidable. Note that the obtained predicate is \( \neg \psi_{\alpha,\beta,M} = \zeta_{\alpha,M} \land \xi_{\beta,M} \land \delta_{\alpha,\beta} \land \neg \phi_{\alpha,\beta,M} \) (or a conjunction of a finite sequence of such predicates if we quantified the elements \( \alpha \) or \( \beta \)).

(ii) To complete the second step we need to compute all contexts in which this negated predicate is true. We show that this is possible by proving efficient computability of the sets of contexts in which every component \( \zeta_{\alpha,M} \), \( \xi_{\beta,M} \), \( \delta_{\alpha,\beta} \), and \( \neg \phi_{\alpha,\beta,M} \) treated separately is true. We also prove that the results are rational sets, hence we can intersect them, obtaining another rational set.

(iii) In the next step we have to intersect the set of contexts obtained in (ii) with the set of all contexts under consideration. However, we do not need to do this explicitly (see (iv)).

(iv) In the case of the problems we are solving, the emptiness (or nonemptiness) of the intersection of the sets described in (iii) can be translated into a Generalised (Set) Reachability Problem.

3 Proof Technique

Let us recall that rational subsets of \( \mathbb{N}^k \) are subsets built from finite subsets with finitely many operations of union \( \cup \), addition \( + \) and star \( ^{*} \).

Theorem 1 (Ginsburg/Spanier [13]). Rational subsets of \( \mathbb{N}^k \) form an effective Boolean algebra (i.e. are closed under union, intersection and difference).

Definition 2 (\( \omega \)-extension). Let \( \mathbb{N}_\omega = \mathbb{N} \cup \{ \omega \} \), where \( \omega \) is a new symbol (of infinity). We extend, in a natural way, the addition operation: \( n + \omega = \omega \), and the order: \( (\forall n \in \mathbb{N}) n < \omega \). The set of \( k \)-dimensional vectors over \( \mathbb{N}_\omega \) will be denoted by \( \mathbb{N}_\omega^k \), its elements are called \( \omega \)-vectors. The addition operation \( + \) and the ordering relation \( \leq \) in \( \mathbb{N}_\omega^k \) are understood componentwise. For \( X \subseteq \mathbb{N}_\omega^k \), we denote by \( \text{Min}(X) \) the set of all minimal (w.r.t. \( \leq \)) members of \( X \), and by \( \text{Max}(X) \) the set of all maximal (w.r.t. \( \leq \)) members of \( X \).

Fact 1 (Dickson [10]) Any subset of incomparable elements of \( \mathbb{N}_\omega^k \) is finite.

Definition 3 (Closures, convex sets, bottom and cover).

- Let \( x \in \mathbb{N}_\omega^k \) and \( X \subseteq \mathbb{N}_\omega^k \). We denote: \( \downarrow x = \{ z \in \mathbb{N}^k | z \leq x \} \), \( \uparrow x = \{ z \in \mathbb{N}^k | x \leq z \} \), \( \downarrow X = \bigcup \{ \downarrow x | x \in X \} \), \( \uparrow X = \bigcup \{ \uparrow x | x \in X \} \), and call the sets left and right closures of \( X \), respectively;
- A set \( X \subseteq \mathbb{N}^k \) such that \( X = \downarrow X \) is said to be left (right) closed;
- A set \( X \subseteq \mathbb{N}^k \) such that \( X = \uparrow X \) is said to be convex;
- The left \( \omega \)-closure of \( X \subseteq \mathbb{N}^k \) is the set \( \downarrow X_\omega = \{ z \in \mathbb{N}^k | \exists z \subseteq X \} \);
- Bottom and Cover of \( X \subseteq \mathbb{N}^k \) are the sets \( \text{Bottom}(X) = \text{Min}(X) \) and \( \text{Cover}(X) = \text{Max}(\downarrow X_\omega) \), resp. We will write \( \text{Bo}X \) and \( \text{Co}X \), for short.

Proposition 1 ([2]). Any convex subset of \( \mathbb{N}^k \) is rational.
In this paper we are interested in particular sets of markings, which are needed to establish the quantifier context in the discussed formulas. Let us define the following sets of markings (for given steps $\alpha$ and $\beta$):

- $E_\alpha = \{ M \in \mathbb{N}^k | M[\alpha] \}$ - markings enabling $\alpha$
- $E_{\alpha \beta} = \{ M \in \mathbb{N}^k | (\exists w \in (2^T)^*)M[w\alpha] \}$ - markings enabling a step sequence $\alpha \beta$
- $E_{\alpha \backslash \beta} = \{ M \in \mathbb{N}^k | (\exists w \in (2^T)^*)M[\alpha \backslash \beta] \}$ - mark. enabling $\alpha \backslash \beta$

The formulation of the expressions for some of the considered sets is immediate.

Let us note the following equalities:

- $E_\alpha = en_\alpha + \mathbb{N}^k$
- $E_{\alpha \beta} = \text{cw}_\text{max}(en_\alpha, en_\alpha - ex_\beta + en_\beta) + \mathbb{N}^k$
- $E_{\alpha \backslash \beta} = \text{cw}_\text{max}(en_\alpha, en_\alpha - ex_\alpha + en(\beta \backslash \alpha)) + \mathbb{N}^k$

where $\text{cw}_\text{max}$ is componentwise maximum (i.e., the resulting vector has the largest value at every coordinate).

**Fact 2** The sets $E_\alpha$, $E_{\alpha \beta}$, $E_{\alpha \backslash \beta}$, are rational.

**Proof.** Note that all those sets are right-closed, hence convex, so, by Proposition 1, rational. $\square$

The expressions for the rest of the sets mentioned above are more complex. That is why we take advantage of the theory of residual sets [22].

**Definition 4** (Valk/Jantzen [22]). A subset $X \subseteq \mathbb{N}^k$ has property RES if and only if the problem “Does $\downarrow v$ intersect $X$?” is decidable for any $\omega$-vector $v \in \mathbb{N}^k$. 

**Theorem 2** (Valk/Jantzen [22]). Let $X \subseteq \mathbb{N}^k$ be a right-closed set. Then Bottom of $X$ is effectively computable if and only if $X$ has property RES.

**Lemma 1.** The following sets have the property RES:

- $E_{\alpha \cdot \beta}$

**Proof.** Clearly, the sets $E_{\alpha \cdot \beta}$, $E_{\alpha \backslash \beta}$, are right-closed, by the monotonicity property. We shall prove that they have the property RES.

Let us notice that $\downarrow v$ intersects $E_{\alpha \cdot \beta}$ if and only if there is a path in the concurrent coverability graph of the net $(P, T, W, v)$ containing an arc labelled by $\alpha$.

Observe also that, $\downarrow v$ intersects $E_{\alpha \backslash \beta}$ if and only if $en_\alpha \leq v$ (i.e. $\alpha$ is enabled at $v$) and there is a path in the concurrent coverability graph of the net $(P, T, W, v')$, where $v'$ is an $\omega$-marking obtained from $v$ by execution of $\alpha$, containing an arc labelled by $\beta$. $\square$

**Lemma 2.** The Bottoms of the sets $E_{\alpha \cdot \beta}$, $E_{\alpha \backslash \beta}$, are effectively computable.

**Proof.** The sets listed above have the property RES (Lemma 1), and hence, by Theorem 2 their Bottom are effectively computable. $\square$
Corollary 1. The sets $E_{\alpha}, E_{\alpha\beta}, E_{\alpha..\beta}$ are rational. Moreover, the rational expression for a set $X$ from the above list is as follows: $X = \text{Bottom}(X) + \mathbb{N}^k$.

In Section 2 we provided a general idea of checking the decidability of formulas depicted in Figure 2. The goal was to find a set of contexts, i.e., the set of markings for which the considered negated formula is true. The computation of such sets of markings becomes possible thanks to the sets listed above ($E_{\alpha}, E_{\alpha\beta}, E_{\alpha..\beta}$). As all of those sets are effectively rational, we can intersect or sum them up obtaining other rational sets. We use this method for computing sets of undesirable markings, i.e., sets of markings at which the examined formula is false (see formulas of Figure 2). Examples of rational expressions for certain formulas are presented in the following section.

As noted in Section 2, the next step is to investigate whether any of the unwanted markings is reachable in a particular net. This task is reduced to decidability of the following problem:

**Generalised (Set) Reachability Problem**

**Instance:** A net $N = (P, T, W, M_0)$ and a set $X \subseteq \mathbb{N}^{|P|}$.

**Question:** Is there a marking $M \in X$, reachable in $N$?

Possessing a rational expression for every unwanted set and having in mind that all the sets are convex, it is enough to check whether any marking from the undesirable set connected to a distinct decision problem is reachable in a given net. The reachability of any undesirable marking gives us a negative answer for problems related to formulas of Figure 2. The following theorem yields decidability of all the problems corresponding to formulas of Figure 2.

**Theorem 3 ([2]).** If $X \subseteq \mathbb{N}^{|P|}$ is a rational convex set, then the Generalised Reachability Problem for $X$ is decidable in the class of p/t-nets.

**Remark 1.** It should be noted that rational sets are exactly semi-linear sets and the reachability of semi-linear sets has already been shown in [15]. However, we refer to the proof of [2], because it fits perfectly into our applications.

4 Applications

4.1 Sequential Semantics

The first approach to the notion of persistence oriented towards weak liveness, not only enabledness of actions, appeared in [2]. One can find there three classes of persistence defined for nets with sequential semantics: the first one (corresponding to the classical notion): “no action can disable another one”, and two generalizations of this notion: “no action can kill another one” and “no action can kill another enabled one”. Let us recall the notions:

**Definition 5 (Three kinds of persistence).**

Let $N = (P, T, F, M_0)$ be a place/transition net. If $(\forall M \in [M_0])(\forall a, b \in T)$
– $M[a] \land M[b] \land a \neq b \Rightarrow M[ab]$, then $N$ is said to be $e/e$-persistent (every enabled action stays enabled after the execution of any other action);
– $M[a] \land (\exists u)M[ub] \land a \neq b \Rightarrow (\exists v \in T^*)M[avb]$, then $N$ is said to be $l/l$-persistent (every weakly live action stays weakly live after the execution of any other action);
– $M[a] \land M[b] \land a \neq b \Rightarrow (\exists v \in T^*)M[avb]$, then $N$ is said to be $e/l$-persistent (every enabled action stays weakly live after the execution of any other action).

In [2] the following decision problems were proved to be decidable:

**Instance:** A $p/t$-net $N = (P,T,F,M_0)$.

**Questions:**

- **EE Net Persistence Problem:** Is the net $e/e$-persistent?
- **LL Net Persistence Problem:** Is the net $l/l$-persistent?
- **EL Net Persistence Problem:** Is the net $e/l$-persistent?

We use an analogous proving technique in this paper. Let us notice that the problems described above correspond to the bottom vertex of the lattice depicted in Figure 2, where we can find the following formula: $\forall M \forall \alpha \forall \beta \psi_{\alpha,\beta,M}$.

**Example 2.** For EL (LL) Net Persistence Problem we fix the framework parameters as follows:

- object type - $\mathcal{A}$ transitions – sequential semantics
- difference – standard difference of elements of the set of actions
- initial enabledness of $\alpha$ - $\mathcal{A}$ immediate standard enabledness of an action $a$ such that $\alpha = \{a\}$
- initial enabledness of $\beta$ - $\mathcal{A}$ immediate standard enabledness (resp. ultimate weak liveness) of an action $b$ such that $\beta = \{b\}$
- terminal enabledness - $\mathcal{A}$ ultimate weak liveness of an action $b$, where $\beta = \{b\}$

Hence, we obtain the following formulas (for EL and LL, respectively):

$\forall M \forall \alpha \forall \beta \psi_{\alpha,\beta,M} \equiv \forall M \forall \alpha=\{a\} \forall \beta=\{b\} M[a] \land M[b] \land a \neq b \Rightarrow \exists \underbrace{v \in T \cdots}_{T^*} M[avb]$, 
$\forall M \forall \alpha \forall \beta \psi_{\alpha,\beta,M} \equiv \forall M \forall \alpha=\{a\} \forall \beta=\{b\} M[a] \land \exists \underbrace{u \in T \cdots}_{T^*} M[ub] \land a \neq b \Rightarrow \exists \underbrace{v \in T \cdots}_{T^*} M[avb]$.

In Figure 3 we present the $p/t$-net $N$ together with its reachability graph. The net $N$ is $e/l$-persistent but not $l/l$-persistent.

The rational expressions for the sets of undesirable markings are as follows: $\bigcup_{\alpha,\beta} E_\alpha \cap E_\beta \cap (\mathbb{N}^{|P|} \setminus E_{\alpha,\beta})$ for $e/l$-persistence, and $\bigcup_{\alpha,\beta} E_\alpha \cap E_\beta \cap (\mathbb{N}^{|P|} \setminus E_{\alpha,\beta})$ for $l/l$-persistence.

It should be noted that the decidability of step oriented versions of the problems above (where pairs of actions were settled) is used in the proofs, which is why the formula $\forall M \psi_{\alpha,\beta,M}$ was also the subject of interest.

**Remark 2.** The EE Net Persistence Problem is the old and classical Net Persistence Problem raised by Karp/Miller [16]. Later Hack [15] showed that it is reducible to the Reachability Problem, which turned out to be decidable (Mayr [20], Kosaraju [17]).
Fig. 3. A p/t-net which is e/l-persistent but not l/l-persistent, and its reachability graph (a kills b indirectly).

Let us notice that the concepts recalled above concern entire nets. A given net is persistent in one of those three meanings, if for every reachable marking of the net and every pair of different transitions a certain condition holds.

In our discussion, however, we would like to study the notion of persistence in a more local sense. We shall consider individual executions of actions at specified markings. This approach leads us to the analysis of two concepts: nonviolence and persistence [18].

Definition 6 (Persistence and nonviolence).
Let \( t \) be a transition enabled at a marking \( M \) of a p/t-net \( N \). Then:

\( t \) is locally nonviolent at \( M \) if, for every \( t’ \) enabled at \( M \) \( t’ \neq t \Rightarrow M[t’t] \)

\( t \) is locally persistent at \( M \) if, for every \( t’ \) enabled at \( M \) \( t’ \neq t \Rightarrow M[t’t] \).

Let us notice, that local notions correspond to formulas \( \forall \beta \psi_{\alpha,\beta,M} \) (nonviolence) and \( \forall \alpha \psi_{\alpha,\beta,M} \) (persistence) in Figure 2.

Definition 7 (Globally nonviolent and persistent nets).
A transition is globally nonviolent (globally persistent) in a p/t-net \( N \) if it is locally nonviolent (locally persistent, resp.) at every reachable marking of \( N \) (at which it is enabled). A p/t-net net is globally nonviolent (globally persistent) if it contains only globally nonviolent (globally persistent, resp.) transitions.

Fact 3 Let \( N = (P, T, F, M_0) \) be a p/t-net. Then the following are equivalent:

\( N \) is globally nonviolent.
\( N \) is globally persistent.
\( N \) is e/e-persistent.

In the next subsection we discuss in details (in step semantics) local properties of persistence and nonviolence.
4.2 Step Semantics

Another approach to the notion of persistence appeared in [11] and [18] uses step semantics. In this case we are interested in steps (sets of transitions) instead of single actions and we specify the step difference, which turns out to be symptomatic. In [18] three significantly different notions of persistence/nonviolence based on different definitions of distinguishability of steps are defined. We start from recalling those three notions:

**Definition 8 (nonviolent and persistent steps).** Let $\alpha$ be a step enabled at a marking $M$ of a p/t-net $N$. Then:

- $\alpha$ is locally $A$-nonviolent ($\beta$ is locally $A$-persistent) at marking $M$ (or $\text{LA}$-nonviolent/$\text{LA}$-persistent) if, for every step $\beta$ ($\alpha$ respectively) enabled at $M$, $\beta \cap \alpha = \emptyset \implies M[\alpha \beta]$ [18],

- $\alpha$ is locally $B$-nonviolent ($\beta$ is locally $B$-persistent) at marking $M$ (or $\text{LB}$-nonviolent/$\text{LB}$-persistent) if, for every step $\beta$ ($\alpha$ respectively) enabled at $M$, $\alpha \setminus \beta \neq \emptyset \neq \beta \setminus \alpha \implies M[\alpha \beta]$,\ 

- $\alpha$ is locally $C$-nonviolent ($\beta$ is locally $C$-persistent) at marking $M$ (or $\text{LC}$-nonviolent/$\text{LC}$-persistent) if, for every step $\beta$ ($\alpha$ respectively) enabled at $M$, $\beta \neq \alpha \implies M[\alpha \beta]$ [18].

A step $\alpha$ (which is weakly live at $M_0$) is globally $A/B/C$-nonviolent/persistent (or $\text{GA/GB/GC}$-nonviolent/persistent) in $N$ if it is respectively $\text{LA/LB/LC}$-nonviolent/persistent at every reachable marking of $N$ at which it is enabled.

However, the decidability problems related to the notions defined this way were stated as future work in [11] and [18] and can be formulated as follows:

**Instance:** A p/t-net $N = (P, T, F, M_0)$.

**Questions:**

- **Step** $\text{GA/GB/GC–Nonviolence Problem:}$
  
  Is the step $\alpha$ globally $A/B/C$-nonviolent in $N$?

- **Step** $\text{LA/LB/LC–Nonviolence Problem:}$
  
  Is there a marking $M \in [M_0]$ such that the step $\alpha$ is locally $A/B/C$-nonviolent at $M$?

- **Step** $\text{GA/GB/GC–Persistence Problem:}$
  
  Is the step $\beta$ globally $A/B/C$-persistent in $N$?

- **Step** $\text{LA/LB/LC–Persistence Problem:}$
  
  Is there a marking $M \in [M_0]$ such that the step $\beta$ is locally $A/B/C$-persistent at $M$?

We rewrite two of those problems in terms of the developed framework. The formulas corresponding to the other two need to introduce existential quantifiers which is out of the scope of this paper. We obtain:

- In the case of Step $\text{GA/GB/GC–Nonviolence Problem:}$ $\forall_M \forall_\beta \forall_\alpha, \beta, M \exists_M \forall_\beta \theta_\alpha, M$.

- In the case of Step $\text{GA/GB/GC–Persistence Problem:}$ $\forall_M \forall_\alpha \forall_\beta, \beta, M \exists_M \forall_\alpha \theta_\beta, M$. 
Note that $\forall M \forall \psi_{\alpha,\beta,M} \text{ and } \forall M \forall \psi_{\alpha,\beta,M}$ are problems depicted in Figure 2. Hence, to prove their decidability, we can use the technique described in this paper. All we need to do is fixing all the framework parameters and find the rational expressions for some sets of markings.

**Example 3.** For Step GA–persistence (LC–nonviolence) Problem we fix the framework parameters as follows:

- object type - steps (step semantics)
- difference - standard disjointness (resp. difference) of the sets of actions
- initial enabledness of $\alpha$ - immediate standard enabledness of a step
- initial enabledness of $\beta$ - immediate standard enabledness of a step
- terminal enabledness - ultimate weak liveness of a step.

Hence, we obtain the following formula for GA–persistence:

$$\forall M \forall \psi_{\alpha,\beta,M} \equiv \forall M \in [M_0] \forall \alpha \in 2^{P(M)} (M_{\alpha} \land M_{\beta} \land \alpha = \emptyset \Rightarrow M_{\alpha \beta})$$

Note that for LC–nonviolence we would deal with the following formula

$$\exists M \in [M_0] \forall \beta \in 2^{P(M_{\alpha} \land \alpha \neq \beta)} \land (M_{\beta} \land M_{\alpha} \land \alpha \neq \beta \Rightarrow M_{\alpha \beta})$$

See Figure 4 for a net which is LC–nonviolent but not GA–persistent. For more discriminating examples see [18].

![Fig. 4. A p/t-net N in which step \{a\} is LC–nonviolent but not GA–persistent.](image)

**Example 4.** Let us we fix the framework parameters as follows:

- object type - steps (step semantics)
- difference - standard disjointness of the sets of actions
- initial enabledness of $\alpha$ - immediate standard enabledness of a step
- initial enabledness of $\beta$ - immediate standard enabledness of a step
- terminal enabledness - ultimate weak liveness of a step.

Note that this is the framework for $A$-(e/l)-nonviolence and $A$-(e/l)-persistence in step semantics (which are (e/l)-nonviolence and (e/l)-persistence, where step difference
means emptiness of the intersection of steps - type \( \Lambda \). Moreover,

\[
\psi_{\alpha,\beta,M} \equiv M[\alpha] \land M[\beta] \land (\alpha \cap \beta = \emptyset) \Rightarrow \exists v \in (2^T), M[\alpha v \beta].
\]

In Figure 5 we put the rational expressions for the sets of undesirable markings concerning formulas from Figure 2. For consistency, we formulate in the same manner expressions for the cases with fixed markings. To check whether the fixed marking \( M \) belongs to a distinct set (of undesirable markings) it is enough to intersect this set with a singleton \( \{M\} \).

Naturally, for given steps \( \alpha, \beta \) and a fixed marking \( M \) the formula \( \psi_{\alpha,\beta,M} \) depicted on top of Figure 2 is easy to verify, regardless of the values of the framework parameters using concurrent coverability graphs.

\[\text{Fig. 5. Lattice of expressions for undesirable markings. Note that in the ranges of all unions we take only } \alpha \cap \beta = \emptyset.\]

Remark 3. The left sides of the diagrams in Figures 2 and 5 correspond to the notion of nonviolence, the right sides are dedicated to persistence. The middle columns of vertices can be associated with the properties of markings or whole nets.

5 Conclusions and Future Work

In this paper we proved decidability of several decision problems concerning persistence notions. We split them into two classes and introduced a unified solution that allows to proceed in the same manner in many cases. The proof technique does not depend on the chosen parameters of the general framework. Besides the classical immediate persistence and nonviolence notions we covered more general weak-live oriented ones.

In particular, starting from classical notions defined in [19], we recalled and reformulated the problems from [2] and proved the decidability of the decision problems left open in [18]. In general, the presented framework allows to switch smoothly to the step semantics.
A natural way of non-trivial extending our framework would be to introduce existential quantifiers. As we already noted, this would allow to deal with local persistence and nonviolence in step semantics.

It is worth mentioning that the notion of weak persistence (see [24, 25]) does not yet fit into the introduced framework. Another notion, not yet covered, is restricted persistence oriented on weak liveness (see the notion of $e/l-k$-persistence from [5]). This shows the natural direction of future development.

In [2] and [18] inclusions between kinds of persistence notions defined there were investigated. It would be interesting examine the relationships between wider spectrum of persistence notions obtaining a detailed taxonomy.

Acknowledgement

We are immensely grateful to Eike Best and Maciej Koutny for their comments and suggestions on earlier versions, although any errors are our own and should not tarnish the reputations of these esteemed persons. We also thank all anonymous reviewers for their notes and remarks. The study is cofounded by the European Union from resources of the European Social Fund. Project PO KL Information technologies: Research and their interdisciplinary applications, Agreement UDA-POKL.04.01.01-00-051/10-00 and by the Polish National Science Center (grant No.2013/09/D/ST6/03928).

References


Specifying Functional Programs with Intuitionistic First Order Logic

Marcin Benke

Institute of Informatics, University of Warsaw

Abstract. We propose a method of specifying functional programs (in a subset of Haskell) using intuitionistic first order logic, that works well for inductive datatypes, higher-order functions and parametric polymorphism.

1 Introduction

Will we ever know the answer to all questions? Can one enter the same river twice? These and similar questions have been asked again and again since classical times and even long before that. Yet, ironically, the classical logic views the world as fixed and cognizable. A model of classical logic is a static structure with complete information about the relations between its elements. Such is the outlook that justifies the infamous tertium non datur: \( p \lor \neg p \).

Software development is one of the disciplines, where one is recurringly, and often painfully reminded that such outlook is not only idealized, but often naïve: there is no one static world, but a multiverse of branching and merging worlds.; not only are we far from having all answers, they are rarely final even when we have them. Indeed there is a lot of bitter truth to the adage that the only constant is change.

Intuitionistic logic offers an attractive option for modelling a world of constant change and incomplete information. A Kripke model consists of a multitude of worlds, connected by an ordering relation. This relation may be time (earlier and later state of the world), but not necessarily so; nor need this order be linear; commit graph in a version control system gives a decent approximation, though we can and sometimes do consider models with infinite number of worlds as well.

One of the selling points of functional programming is its potential for easier and better specification and verification. While this potential is indisputable, the tools and methods to realise it are still lacking (see e.g. [1]).

Most approaches use either first order classical logic (e.g. [2, 6]) or higher-order logic (e.g. [4]). In this paper we propose a method of specifying Haskell programs using intuitionistic first order logic, that works well for inductive datatypes, higher-order functions and parametric polymorphism.

* This work has been supported by Polish NCN grant NCN 2012/07/B/ST6/01532.
2 The Logic

2.1 Core Logic

Our logic is essentially a variant of $\lambda P$ from [5] extended with constructs for existential quantifiers, alternative, conjunction and falsity. There are no separate constructs for implication or negation, as these can be easily encoded.

$$
\begin{align*}
\Gamma & ::= \{ \} \mid \Gamma (x : \phi) \mid \Gamma (\alpha : \kappa) \\
\kappa & ::= * \mid (\Pi x : \phi)\kappa \\
\phi & ::= \alpha \mid (\forall x : \phi)\phi \mid (\exists x : \phi)\phi \mid \phi \land \phi \mid \phi \lor \phi \mid \bot \\
M & ::= x \mid (\lambda x : \phi.M) \mid (M_1 M_2) \mid [M_1, M_2]_{\exists x : \phi, \phi} \mid \\
& \quad \text{abstract} (x : \phi_1, y : \phi_2) = M_1 \text{ in } M_2 \mid (M_1, M_2)_{\phi_1 \land \phi_2} \mid \\
& \quad \pi_1M \mid \pi_2M \mid \text{in}_1, \phi_1, \text{phi}_2 M \mid \text{in}_2, \phi_1, \lor \phi_2 M \mid \\
& \quad \text{case } M_1 \text{ in } (\text{left } x : \phi_1.M_2)(\text{right } y : \phi_2.M_3) \\
& \quad \varepsilon_\phi (M)
\end{align*}
$$

2.2 Notation and Extensions

Additional connectives

\[ a \rightarrow b \equiv \forall a.b \]
\[ a \leftrightarrow b \equiv a \rightarrow b \land b \rightarrow a \]
\[ -a \equiv a \rightarrow \bot \]

A Universe of values $V : *$ is assumed. Quantifiers usually range over this universe, hence

$$
\forall x.\phi \equiv \forall x : V.\phi
$$

Axiom schemas of the form

\[ \text{schema name}(P:\text{kind}) : \text{formula} \]

are to be understood as a finite set of formulas: one for every predicate symbol of the appropriate kind in the signature.

Proof sketches are rendered in a Mizar-like notation (cf. e.g. [3, 8, 7])

Haskell code is written using so called “Bird-tracks”, e.g.

\[ > \text{id} :: a \rightarrow a \]
\[ > \text{id } x = x \]

Note The approach described in this document is “untyped” in the sense that we don’t use Haskell type declarations, but derive our own types. Hence we might call our approach “owntyped” (there is also a strong connection with refinement types). On the other hand, we still use data type declaration as a source of useful information.
3 Datatypes

In this section we illustrate our method on some example Haskell datatypes and functions, starting with the simplest ones and progressing towards more complex ones.

3.1 Bool

> data Bool = False | True

We can characterize Bool by the following axiom

axiom defBool : \( \forall x. \text{Bool}(x) \leftrightarrow x=\text{False} \lor x=\text{True} \)

or by an axiom schema

schema elimBool(P):
(P(False) \land P(True)) \rightarrow \forall x. \text{Bool}(x) \rightarrow P(x)

Now consider the following definition

bnot False = True
bnot True = False

This definition can be characterized as follows

axiom defBnot : bnot False = True \land bnot True = False

Now let’s prove that not takes Bool to Bool (in Mizar-like notation):

theorem typeBnot : \( \forall x. \text{Bool}(x) \rightarrow \text{Bool}(\text{bnot } x) \)

proof
consider x st Bool(x)
then x = False \lor x = True by defBool
thus thesis by cases
  suppose x = False
  then bnot x = True
  thus thesis
  suppose x = True
  then bnot x = False
  thus thesis
end

An alternative proof of typeBnot, using elimBool

theorem typeBnot : \( \forall x. \text{Bool}(x) \rightarrow \text{Bool}(\text{bnot } x) \)

proof
consider x st Bool(x)
\text{Bool}(\text{True}) \land \text{Bool}(\text{False}) by defBool
then \text{Bool}(\text{bnot False}) \land \text{Bool}(\text{bnot True}) by defBnot
let P(x) = Bool(bnot x)
thus thesis by elimBool(P)
end
This seems like an overkill and can probably be proved automatically. However, note that our statement is substantially stronger than a simple type assertion: it also states that \texttt{bad} terminates for all inputs. Now, what about a function that doesn’t? Consider

\begin{verbatim}
bad True = True
bad False = bad False
\end{verbatim}

In Haskell, \texttt{bad :: Bool \rightarrow Bool}, but a theorem like

\begin{verbatim}
\forall x.\text{Bool}(x) \rightarrow \text{Bool}(\text{bad } x)
\end{verbatim}

is not provable. On the other hand, we can prove

\begin{verbatim}
theorem notSoBad :
\forall x.(\text{Bool}(x) \land x \neq False)
\rightarrow \text{Bool}(\text{bad } x)
\end{verbatim}

### 3.2 Nat

\begin{verbatim}
> data Nat where \{ Z :: Nat; S :: Nat \rightarrow Nat \}
\end{verbatim}

\texttt{env Z, S : V}

\texttt{axiom introNat : Nat(Z) \land \forall n. Nat n \rightarrow Nat (S n)}

\texttt{schema elimNat (P:V\rightarrow*) =}

\begin{verbatim}
( P Z \\
& \forall n. Nat n \rightarrow P n \rightarrow P(S n) \\
) \rightarrow \forall m. P m
\end{verbatim}

Alternative (and equivalent?) elimination

\texttt{schema elimNat (P:V\rightarrow*) =}

\begin{verbatim}
( P Z \\
& (\forall n. P n \rightarrow P (S n)) \\
) \rightarrow \forall m. P m
\end{verbatim}

Now we can define some functions

\begin{verbatim}
> plus Z x = x \\
> plus (S n) x = S(plus n x)
\end{verbatim}

\texttt{axiom plusDef : \forall x.plus Z x = x}

\texttt{\land \forall n x.plus (S n) x = S(plus n x)}

Some properties
theorem plusType : \forall x y. Nat(x) \rightarrow Nat(y) \rightarrow Nat(plus x y)
proof
\forall y. plus Z y = y by plusDef
then \forall y. Nat(y) \rightarrow Nat(plus Z y)
\forall n x. Nat(plus n x) \rightarrow Nat(S (plus n x)) by introNat
then \forall n x. Nat(plus n x) \rightarrow Nat(plus (S n) x) by plusDef
thus thesis by elimNat(P) where
P n = \forall y. Nat(plus n y)
end

predicate PlusZ(n : V) = plus x Z = x
theorem plusZR : \forall n. Nat(n) \rightarrow plusZ(n)
proof
plus Z Z = Z by plusDef
\forall n. plus n Z = n -> S(plus n Z) = S n by equality
\forall n. plus (S n) Z = S(plus n Z) by plusDef
then \forall n. plus n Z = n -> plus (S n) Z = S n
thus thesis by elimNat(plusZ)
end

3.3 Lists

To avoid confusion, we write the list type as List a and the corresponding predicate as List rather than use the usual [a]. In practice this is just a matter of syntactic sugar.

> data List a = Nil | Cons a (List a)

Lists can be axiomatised as follows:

env List : V\rightarrow* \rightarrow *, Nil : V, Cons : V \rightarrow V
schema introList(T:V\rightarrow*)
  = List(T)([])
  & (T(x)&List(xs) \rightarrow List(Cons x xs))
schema elimList(T,P:V\rightarrow*)
  = P(Nil)
  & (\forall x xs. T(x) & P (xs) \rightarrow P(Cons x xs))
  \rightarrow \forall xs. List(T) (xs) \rightarrow P(xs)

Sample theorem for map

> id x = x
> map f Nil = Nil
> map f (Cons x xs) = Cons (f x) (map f xs)

axiom mapDef : map f Nil = Nil & \forall f x xs...
theorem mapType(T,U: V\rightarrow*) : (\forall x. T(x) \rightarrow U(f x))
\[
\rightarrow (\forall \text{xs}. \text{List}(T)(\text{xs}) \rightarrow \text{List}(U)(\text{map f \text{xs}}))
\]

**theorem mapId(T:V\rightarrow*)** : \(\forall \text{xs}. \text{map id \text{xs}} = \text{xs}\)

**proof**

let \(P(\text{xs}:V) = \text{map id \text{xs}} = \text{xs}\)

have \(P(\text{Nil}) \& \forall x \quad x \cdot P(x) \rightarrow P(\text{Cons x \text{xs}})\) by mapDef

thus thesis by elimList(P)

Consider a (slightly convoluted) example of a function summing a list:

```
sum :: List Nat -> Nat
sum Nil = Z
sum (Cons n ns) = case n of
  Z -> sum (Cons n ns)
  (S m) -> S(sum (Cons m ns))
```

this can be characterized as follows:

```
axiom sumNil : sum Nil = Z
axiom sumCons :
  \forall n ns. (n = Z \rightarrow sum (Cons n ns) = sum ns)
  \& (\forall m. n = S m
  \rightarrow sum (Cons n ns) = S(sum (Cons n ns))
```

## 4 Polymorphic Functions

If types translate to predicates, then one might think quantification over types might requiring quantifying over predicates. But we may avoid this reading “for all types \(a\) and values \(x\) of type \(a\)” as simply “for all \(x\) (regardless of type)”.

```
const :: a -> b -> a
const x y = x
--# axiom forall x y. const x y = x
```

## 5 Conclusions and Future Work

We have proposed a method of specifying Haskell programs using intuitionistic first order logic, that works well for inductive datatypes, higher-order functions and parametric polymorphism. On the other hand, one big remaining challenge is handling also ad-hoc polymorphism, i.e. type classes. One idea we’ve toyed with went along the following lines (in a notation slightly different to what we have used so far):

```
class Functor f where
  fmap :: forall a b.(a->b) -> f a -> f b
  -- fmap_id :: forall a.f a -> Prop
```
instance Functor Maybe where
    fmap f Nothing = Nothing
    fmap f (Just x) = Just f x

-- # axiom Functor_im(Nothing)
-- # axiom forall x. Functor_im(Just x)

-- # conjecture forall i. Functor_im(i) -> fmap id i = i

This is not yet completely satisfactory and needs more work.

References

Exploration of Knowledge Bases Inspired by Rough Set Theory

Agnieszka Nowak-Brzezinska and Alicja Wakulicz-Deja

Department of Computer Science, Institute of Computer Science, Silesian University, Poland
http://ii.us.edu.pl

Abstract. In this paper, we discuss some issues related to exploration of knowledge bases inspired by the rough set theory, which emerged 30 years ago, and is nowadays a rapidly developing branch of artificial intelligence. The partition of rules approach allows us to divide a large set of rules into smaller subsets that are easier to manage. Optimisation relies on reducing the number of rules searched in each run of inference. The paper presents the definition of the knowledge base model based on partition of rules and a modification of the forward inference algorithm for groups of rules generated by the partition strategy. It also contains a simple case study with an example of the partition of rules and the inference process for a simple knowledge base as well as experimental results.

Key words: knowledge base, inference process, goal driven, data driven, decision support system

1 Introduction

For the last twenty years, there has been an enormous interest in integrating database and knowledge-based system technologies to create an infrastructure for modern advanced applications. The result of it is a knowledge base (KB) system which consists of database systems extended with some kind of knowledge, usually expressed in the form of rules\(^1\) - logical statements (implications) of the type "if condition\(_1\) \& \ldots \& condition\(_n\) then conclusion". The knowledge of experts, expressed in such a natural way, makes rules easily understood by people not involved in the expert system building. KBs are constantly increasing in volume, thus the knowledge stored as a set of rules is getting progressively more complex and searching such sets is an important data-mining task. When rules are not organized into any structure, the system is inefficient. There is a growing research interest in searching for methods that manage large sets of rules. Most of them use the clustering approach as well as joining and reducing rules [1, 2, 10, 11]. This paper presents a different idea, in which rules are divided into a number of groups based on similar conditions and/or conclusion. The process is called partition of rules\(^2\) and is conducted by so-called partition strategies. In the

\(^1\) Rules have been extensively used in knowledge representation and reasoning. It is very space efficient: only a relatively small number of facts needs to be stored in the KB and the rest can be derived by the inference rules.

\(^2\) The idea is new but it is based on the authors’ previous research, where the idea of clustering rules as well as creating so-called decision units was introduced[9, 5].
authors’ opinion, partition of rules leads to the improvement of the inference process efficiency [5]. If we assume that the user wants to get an answer from the system as soon as possible, the proposed modification of the KB structure reduces the time of inference. Instead of searching within the whole set of rules (as in case of traditional inference processes), only representatives of groups are compared with the set of facts and/or hypothesis to be proven. The most relevant group of rules is selected and the exhaustive searching is done only within a given group. We show how exploration of complex KBs can be applied based on the partition of rules. Our motivation is two-fold. First, we are concerned with how to reason KBs effectively with a large number of rules. Second, we are concerned with improving the efficiency of reasoning over a set of rules by partitioning the set with respect to some strategy. To this end, we provide algorithms for partitioning and reasoning with such partition of rules.

The idea of partition of rules is implemented in the kbExplorer system3, which is not limited to the inference optimization. The practical goal of the project is to create an expert system shell that allows for flexible switching between different inference methods based on knowledge engineer preferences4.

The rest of the paper is organized as follows. In Section 2, the research background is introduced. It also contains a very general description of the proposed concept. Section 3 presents the basic definitions of the partition of rules and partition strategies, as well as the specification of partition’s representative. It also includes the inspiration of the rough set theory. Section 4 describes the forward inference algorithm for the partition of rules which needs to modify the classical algorithm. The results of experiments are given in Section 5. A simple case study is also given. Finally, Section 6 concludes the paper.

2 Related Work and the Proposed Idea

There is a number of studies related to knowledge matching and rules modularization. Interesting and effective approaches to the problem with managing a large set of rules and necessity of its partitioning can be found in [1–3, 10, 11], where known systems like CHIRA, XTT2 or D3RJ are described. In [2] the authors present CHIRA - an algorithm performing decision rules aggregation. Rules are joined if their conditional parts are built from the same conditional attributes or if the conditional attributes set of one rule is a subset of the conditional attributes set of the second one. Another joining algorithm, which operates on rules determined by the dominance-based rough set model, was proposed in [3]. Rules that lie close to each other are joined if their joint causes no deterioration of accuracy of the obtained rule. XTT2 provides a modularized rule base, where rules working together are placed in one context corresponding to a single

3 http://kbexplorer.ii.us.edu.pl/

4 The user has the possibility of creating KBs using a special creator or by importing a KB from a given data source. The format of the KBs enables working with a rule set generated automatically based on the RST theory as well as with rules given apriori by the domain expert. The KB can have one of the following file formats: XML, RSES, TXT. It is possible to define attributes of any type: nominal, discrete or continuous. There are no limits for the number of rules, attributes, facts or the length of the rule.
D3RJ is a method which produces more general rules, where each descriptor can endorse subset of values \[11\]. In most of these tools, a global set of rules is partitioned by the system designer into several parts in an arbitrary way.

The idea of partition of rules, presented in this paper, is based on the previous authors’ research concerning rules clusters and decision units \[8, 9, 5\]. The proposed algorithm is based on the same idea as the classical inference algorithm, but it uses groups’ representatives (playing a role of general rules) instead of each single rule. If a representative matches the searched data, it means that its group probably contains a rule or rules that we are looking for. In this context it is necessary to rewrite the pattern matching algorithm in one point. Instead of finding rules that match exactly a given set of facts, we compare facts with rules’ representatives and the most similar group of rules is selected. Further research is performed only on this group. The advantage is as follows: having \( n \) rules divided into \( k \) groups, only \( k \) groups’ representatives are searched, while in the classical version of the inference process all \( n \) rules have to be analysed.

3 Partition of Rules Idea

The proposed concept assumes division of a KB into coherent subgroups of rules. Therefore, this section presents the basic concepts and notations of KB partitioning. We assume that a KB is a single set \( \mathcal{R} = \{r_1, \ldots, r_n\} \) without any order of \( n \) rules. Each rule \( r_i \in \mathcal{R} \) is stored as a Horn’s clause defined as: \( r_i : p_1 \land p_2 \land \ldots \land p_m \rightarrow c \), where \( p_s \) is \( s \)-th literal (a pair of an attribute and its value) \((a, v^a_s) (s = 1, 2, \ldots, m)\). Attribute \( a \in A \) may be a conclusion of rule \( r_i \) as well as a part of the premises. For every KB with \( n \) rules, the number of possible subsets is \( 2^n \). Any arbitrarily created subset of rules \( \mathcal{R} \subseteq 2^\mathcal{R} \) is called partition of rules (PR) and it can be generated by one of the many possible partition strategies (PS).

3.1 Definition of Partition of Rules

The partition of rules \( PR = \{R_1, R_2, \ldots, R_k\} \), where \( k \) is the number of groups of rules in partition \( PR \), \( R_j \) is \( j \)-th group of rules for \( j = 1, \ldots, k \) and \( PR \subseteq 2^\mathcal{R} \) is generated by one of many possible partition strategies. We need a function \( mc: \mathcal{R} \times PR \rightarrow [0..1] \) to decide whether rule \( r_i \) belongs to group \( R_j \) or not. It is defined individually for every PS. If there is no doubt that rule \( r_i \) belongs to group \( R_j \), \( mc(r_i, R_j) = 1 \). Otherwise, \( mc(r_i, R_j) = 0 \). It means that rule \( r_i \) definitely does not belong to group \( R_j \). Values from the range 0 to 1 mean partial membership. In mathematical meaning, a partition of rules is a collection of subsets \( \{R_j\}_{j \in J} \) of \( \mathcal{R} \) such that: \( \mathcal{R} = \bigcup_{j \in J} R_j \) and if \( j, l \in J \) and \( j \neq l \) then \( R_j \cap R_l = \emptyset \). The subsets of rules are non-empty and every rule is included in one and only one of the subsets\(^5\).

\(^5\) However, we also consider the case, in which a given rule belongs to more than one group. In our future work we are going to extend the definition of the partition in this context.
3.2 Rough Set Theory and Indiscernibility Relation as Inspiration for Partition of Rules Idea

One of the tools for exploration of large rules data sets is analysing the similarities between rules. It leads naturally to their grouping and integration. The similarities of objects are examined by the indiscernibility relations used in the rough set theory (RST)[12, 13]. If \( \mathcal{R} \) is the set of all rules in the \( \mathcal{KB} \), and \( r_i, r_f \) are arbitrary rules, they are said to be indiscernible by \( \mathcal{PR} \), denoted \( r_i \sim \mathcal{PR} r_f \), if and only if \( r_i \) and \( r_f \) have the same value on all elements in \( \mathcal{PR} \):

\[
IND(\mathcal{PR}) = \{ (r_i, r_f) \in \mathcal{R} \times \mathcal{R} : \forall a \in \mathcal{PR}, a(r_i) = a(r_f) \}.
\]

As such, it induces a partition of \( \mathcal{R} \) generated by \( \mathcal{PR} \), denoted \( \mathcal{PR}^* \). Let us assume that the \( \mathcal{KB} \) contains the following rules:

- \( r_1 : (a, 1) \rightarrow (b, 1) \)
- \( r_2 : (a, 1) \rightarrow (c, 1) \)
- \( r_3 : (d, 1) \rightarrow (e, 1) \)
- \( r_4 : (d, 1) \rightarrow (f, 1) \)
- \( r_5 : (b, 1) \rightarrow (g, 1) \)
- \( r_6 : (c, 1) \rightarrow (g, 1) \)
- \( r_7 : (e, 1) \rightarrow (h, 1) \)
- \( r_8 : (f, 1) \rightarrow (h, 1) \)
- \( r_9 : (g, 1) \rightarrow (i, 1) \)
- \( r_{10} : (i, 1) \rightarrow (k, 1) \)
- \( r_{11} : (i, 1) \rightarrow (l, 1) \)
- \( r_{12} : (k, 1) \rightarrow (h, 1) \rightarrow (j, 1) \)
- \( r_{13} : (j, 1) \rightarrow (l, 1) \)
- \( r_{14} : (a, 1) \land (j, 1) \rightarrow (b, 1) \)
- \( r_{15} : (a, 1) \land (j, 1) \land (c, 2) \rightarrow (b, 1) \)
- \( r_{16} : (a, 1) \land (j, 1) \land (c, 2) \land (e, 2) \rightarrow (b, 1) \)
- \( r_{17} : (a, 1) \land (j, 1) \land (c, 2) \land (e, 2) \rightarrow (b, 1) \)

For this \( \mathcal{KB} \) it is possible to create different partitions in terms of different criteria.

An equivalence relation on \( \mathcal{PR} \), defined by premises of rules in \( \mathcal{R} \), is as follows: \( \{ \mathcal{PR} \}^* = \{ r_{13}, r_{16}, r_{15}, r_{14}, r_2, r_1, r_5, r_6, r_7, r_8, r_9, r_{12}, r_4, r_3, r_{11}, r_{10} \} \) while the equivalence relation on \( \mathcal{PR} \) being conclusions of rules produces the partition: \( \{ \mathcal{PR} \} = \{ r_{17}, r_{16}, r_{15}, r_{14}, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{12}, r_4, r_3, r_{11}, r_{10}, r_{13} \} \)

whereas the equivalence relation on \( PR \) for \( PR = (b, 1) \) in conclusions is as follows: \( \{ PR \} = \{ r_{17}, r_{16}, r_{15}, r_{14}, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{12}, r_4, r_3, r_{11}, r_{10}, r_{13} \} \). This last example partition needs additional comments. Usually, when the partition strategy divides rules into the groups that match a given condition (in this case the literal \((b, 1)\)), the final partition is given by the so-called selection in which the first part of the partition is the group that matches a given condition, and the second group does not meet this condition.

3.3 Partition Strategies

Partition strategy is the general concept. It is possible to point out a number of different approaches for creating groups of rules. The most general division of partition strategies talks about two types of strategies: simple and complex. Simple strategies\(^8\) allocate

\(^6\) It does not necessarily contain a subset of attributes. It may also include the criterion of creating groups of rules, i.e. groups of rules with at least \( m \) number of premises or groups of rules with a premise containing a specific attribute or particular pair (attribute, value).

\(^7\) It does not necessarily contain a subset of attributes. It may also include the criterion of creating groups of rules, i.e. groups of rules with at least \( m \) number of premises or groups of rules with a premise containing a specific attribute or particular pair (attribute, value).

\(^8\) It allows for partitioning the rules using the algorithm with time complexity not higher than \( O(nk) \), where \( n = |R| \) and \( k = |PR| \). Simple strategies create final partition \( PR \) by a
every rule \( r_i \) to the proper group \( R_j \), according to the value of the function \( mc(r_i, R_j) \). The example is the strategy of finding a pair of rules the most similar in a given context, i.e. premises of rules. Complex strategies usually do not generate the final partition in a single step. It is defined by a sequence of simple strategies or a combination of them, or by iteration of a single simple strategy. An example is the strategy of creating similarity based partition, which stems from the method of cluster analysis used for rules clustering. It uses a simple strategy which finds pairs of the most similar rules many times. The process terminates if the similarity is no longer at least \( T \). In effect, we get \( k \) groups of rules \( R_1, R_2, \ldots, R_k \) such that \( \bigwedge_{r_i, r_j \in R_k} \text{sim}(r_i, r_j) \geq T \). Each group \( R_i \) contains rules for which mutual similarity is at least equal to \( T \). It is possible to obtain many different rules partitions. In this strategy, rules that form the same group are said to have the same or similar premises. It uses the similarity function \( \text{sim}(r_i, r_j) \): \( \mathcal{R} \times \mathcal{R} \rightarrow [0..1] \) which can be defined in a variety of ways, i.e. based on the conditional part of the rules as

\[
\text{sim}(r_i, r_j) = \frac{|\text{cond}(r_i) \cap \text{cond}(r_j)|}{|\text{cond}(r_i) \cup \text{cond}(r_j)|},
\]

where \( \text{cond}(r_i) \) denotes the conditional part of rule \( r_i \) and \( \text{concl}(r_i) \) its conclusion - respectively. The value of \( \text{sim}(r_i, r_j) \) is equal to 1 if rules are formed by the same literals and 0 when they do not have any common literals. The more similar the rules are the closer to 1 is the value of \( \text{sim}(r_i, r_j) \).

### 3.4 Similarity-Based Partition of Rules

The similarity-based partition strategy, described shortly above, allows for improving the efficiency of the inference process (see alg. Alg01). In the first step of the algorithm, a specific partition (createSingletonGroups), in which every rule forms a separate group, is created: \( \forall R \in PR \mid |R| = 1 \ (\forall R \in PR \mid \bigcup \{ R : R \in PR \}) = n \) and \( \forall R, R_i, \in PR, i \neq j \ R_i \cap R_j = \emptyset \). Further, it finds a pair of the most similar rules iteratively and join them into one group. Similarity is determined by using function \( \text{sim} \) based on the similarity of the conditional part of rules (or groups). This strategy is used to build the partition of 17 rules (presented in section 3.2). At the beginning, each rule creates a single group: \( R_1, \ldots, R_{17} \). Next, the following groups are created: \( R_{18} = \{ R_{17}, R_{16} \}, R_{19} = \{ R_{18}, R_{15} \}, R_{20} = \{ R_{19}, R_{14} \}, R_{21} = \{ R_1, R_2 \}, R_{22} = \{ R_3, R_4 \}, R_{23} = \{ R_{10}, R_{11} \}, R_{24} = \{ R_{20}, R_{13} \}, R_{25} = \{ R_{24}, R_{21} \} \). This algorithm is based on the classical AHC algorithm discussed in [7,9,5]. The partition of rules achieved in this way has got the hierarchical structure thus it is necessary to search the tree of groups of rules.

**Alg01: Similarity-based partition - an algorithm**

**Require:** \( \mathcal{R}, \text{sim}, T \);

*single search of rules set \( R \) according to the value of \( mc(r_i, R_j) \) function described above.*

*For complex strategies time complexity is rather higher than any simple partition strategy.*

9 Let us assume that threshold value \( 0 \leq T \leq 1 \) exists.

10 The time complexity of this operation is \( O(\log n) \) where \( n \) is the number of elements in the tree.
Ensure: \( PR = \{ R_1, R_2, \ldots, R_k \} \);

\[ \text{procedure createPartitions}(R, \text{var } PR, \text{sim}, T) \]
\[ \text{var } R_i, R_j; \]
\[ \begin{align*}
PR &= \text{createSingletonGroups}(R); \\
R_i &= R_j = \{ \}; \\
\text{while findTwoMostSimilarGroups}(\text{sim}, R_i, R_j, PR) \geq T \text{ do} & \quad \text{PR} = \text{rearrangeGroups}(R_i, R_j, PR); \\
\text{end while}
\end{align*} \]
\[ \text{end procedure} \]

3.5 Partition’s Representatives - Profiles

The efficiency of the partition of rules and its usage in the inference process depends on the quality of the created partition. It is based on both: the cohesion and separation of the created groups of rules as well as on the representatives of the groups (Profiles). There can be found many different approaches to determine representatives for groups of rules. There are many possible forms of Profile\((R)\).\(^{11}\) We propose an approach inspired by the RST with lower and upper approximations\(^{12}\). It allows for defining two representatives: Profile\((R_j)\) and Profile\((R_j)\) for every group \(R_j\). The lower approximation set of a profile of group \(R_j\), is the union of all these literals from premises of rules, which certainly appear in \(R_j\), while the upper approximation is the union of the literals that have a non-empty intersection with the subset of interest. As it was mentioned above, rule \(r_i : p_1 \land p_2 \land \ldots \land p_m \rightarrow c\) has a conjunction of \(m\) literals in the conditional part. Each literal \(p_c \in \text{cond}(r_i)\) is a premise of rule \(r_i\). Thus,

\[
\text{Profile}(R_j) = \{ \bigcap_i \text{cond}(r_i) \}
\]

denotes the set of all literals \(p_c\) that are an intersection of the conditional part of each rule \(r_i\) in group \(R_j\), whereas

\[
\text{Profile}(R_j) = \{ \bigcup_i \text{cond}(r_i) : \text{cond}(r_i) \cap \bigcap_f \text{cond}(r_f) \neq \emptyset \}
\]

for \(f \neq i, r_i, r_f \in R_j\) contains the set of all premises of rules creating group \(R_j\) which have a non-empty intersection with the premises of other rules from the same group. For example, if in the KB presented above, group \(R_{22}\) contains two rules \(r_3\) and \(r_4\)

\(^{11}\) It may be (i) the set of all premises of rules that form group \(R\), (ii) the conjunction of the selected premises of all rules included in a given group \(R\) as well as (iii) the conjunction of the premises of all rules included in a given group \(R\).

\(^{12}\) If \(X\) denotes a subset of universe element \(U\) \((X \subset U)\) then the lower approximation of \(X\) in \(B\) \((B \subseteq A)\) denoted as \(BX\), is defined as the union of all these elementary sets which are contained in \(X\): \(BX = \{ x_i \in U | x_i|_{\text{IND}(B)} \subset X \}\). Upper approximation \(BX\) is the union of these elementary sets, which have a non-empty intersection with \(X\): \(BX = \{ x_i \in U | x_i|_{\text{IND}(B)} \cap X \neq \emptyset \}\).
(groups $R_3$, $R_4$), \( \text{Profile}(R_{22}) = \{(d,1)\} \) whereas \( \text{Profile}(R_{22}) = \{(d,1),(b,2)\} \). The \( BN(\text{Profile}(R_{22})) = \text{Profile}(R_{22}) - \text{Profile}(R_{22}) = \{(b,2)\} \). It means that literal \((d,1)\) appears in every rule in this group while \((b,2)\) makes rule \( r_3 \) distinct from \( r_4 \). In other words, \((d,1)\) certainly describes every rule in group \( R_{22} \) while \((b,2)\) possibly describes this group (there is at least one rule which contains this literal in the conditional part). Lower and upper approximations of the profile of a given group are very convenient in further searching. When we want to find a rules which certainly contains some literal, we have to match it to the lower approximation of the profile for every group. If we look for a rule that possibly contains some literal, we only have to match it to the upper approximation of the profiles of groups.

4 Inference Algorithm

In authors’ opinion, both the proposed idea of the partition of rules and high quality of representatives’ representation lead to improve the efficiency of the inference process. There are two inference algorithms: forward and backward\(^\text{13}\). Due to the limited size of the paper only the forward inference algorithm is discussed. The goal of the forward inference process is to find a set of rules with given data included in the premise part of a particular rule. The shorter the time needed to perform such a process, the better. During the inference process, premises of each and every rule are analysed to match them with the current set of facts (\( F \)). Then the subset of the conflict set is chosen consisting of the rules that are actually activated. Finally, previously selected rules are analysed and the set of facts is updated. The most time consuming procedure searches rules and finds those relevant to the given data (goal/facts) \[4\]. The aim is to reduce this process in the best way. Modification of the classical inference algorithm, proposed by the authors, is based on changes in the structure of the rules set. It is no longer a list of all rules. Now it forms a partition of rules with representatives. Thanks to this, the searching procedure is optimized. It needs less time to find the group and then the rule that match a given set of facts.

4.1 Forward Inference Algorithm Based on Partition of Rules

The forward inference algorithm based on the partition of rules (see Alg02) reduces search space by choosing only rules from a particular group of rules, which matches the current facts set. Thanks to the similarity-based partition strategy as the result we get groups of similar rules. In every step, the inference engine matches facts to groups’ representatives and finds a group with the greater similarity value. To find a relevant group of rules, during the inference process, the maximal value of similarity between the set of facts \( F \) (and/or hypothesis to be proven) and the representatives of each group

\(^{13}\) In case of backward inference we look for rules with a given hypothesis as a conclusion \[6\]. Examples of applying the inference in rule \( K Bs \) are described in \[8\], where the modification of the backward inference algorithm for rule \( K Bs \) is presented.
of rules, \( \text{Profile}(R_j) \) is searched\(^{14}\). The more facts with hypothesis are included in the profile of group \( R_j \), the greater the value of similarity. Exhaustive searching of rules is done only within the selected group. At the end, rules from the most promising group are activated. The inputs are: \( PR \) - groups of rules with the representatives and \( F \) - the set of facts. The output is \( F \) the set of facts, including possible new facts obtained through the inference. The algorithm uses temporary variable \( R \), which is the set of rules that is the result of the previous selection.

**Alg02: Modified forward inference algorithm**

**Require:** \( PR, F \);

**Ensure:** \( F \);

```plaintext
procedure forwardInference( PR, var F )
begin
R := selectBestFactMachingGroup( PR, F );
while \( R \neq \emptyset \) do
r := selectRule( R, strategy);
activeRule( r );
R := selectBestFactMachingGroup( PR, F );
end while
end procedure
```

`selectBestFactMachingGroup` is the procedure responsible for finding the most relevant group of rules. If the result of the selection is not empty (\( R \neq \emptyset \)), the `selectRule` procedure is commenced. It finds a rule, in which the premises part is fully covered by facts. If there is more than one rule, the strategy of the conflict set problem plays a significant role in the final selection of one rule. The `activeRule` procedure adds a new fact (conclusion of the activated rule) to the \( KB \). Of course, it is necessary to remove the activated rule from further searching. Afterwards, the `selectBestFactMachingGroup` procedure is called again. The whole presented process is repeated in a while loop until the selection becomes empty. Among the advantages of the proposed modification of the classical forward inference process are: reducing the time necessary to search within the whole \( KB \) in order to find rules to activate and achieving additional information about fraction of rules that match the input knowledge (the set of facts). It is worth to mention that such a modification led to firing not only certain rules but also the approximate rules for which the similarity with the given set of facts is at least equal to \( T \) or is greater than 0.

5 Experiments

The main goal of the research is to show that the inference process for a \( KB \) with the partition of rules is carried out in a shorter time than the classical inference process for a \( KB \) with a list of rules. To do this, it is necessary to compare the following parameters:

\[^{14} sim(F, R_j) = \frac{|F \cap \text{Profile}(R_j)|}{|F \cup \text{Profile}(R_j)|}. \text{The value of } sim(F, \text{Profile}(R_j)) \text{ is equal to 0 if there is no such fact } f_i \text{ (or hypothesis to prove) which is included in the representative of any group } R_j. \text{ It is equal to 1 if all facts (and/or hypothesis) are included in } \text{Profile}(R_j) \text{ of group } R_j.\]
inference time for both inference algorithms, the size of the KB (the number of rules or groups of rules) and the % of the KB that is really analyzed. The results of the inference process on two different structures of the KB (rules and partition of rules) should not differ. Facts extracted from rules should be the same as facts extracted from the partition of rules. Checking this became an additional goal of the experiments. For a better understanding of the inference process for the partition of rules below we show a simple case study, and then the results of the experiments with some interpretation.

5.1 A Simple Case Study

To illustrate an idea of knowledge exploration from the partition of rules, let us consider an example of a KB presented in section 3.2[15]. The classical inference algorithm requires searching each of 17 rules in this set. Using the strategy, presented in section 3.1, based on similar premises only representatives of 8 created groups of rules are analysed. The more rules the greater the profit from using this approach. Usually k << n. The rules partitions, intended for the forward inference are represent by profiles, as follows:

- \( R_5 = \{r_5\}, \text{Profile: } \{(b, 1)\} \)
- \( R_6 = \{r_6\}, \text{Profile: } \{(c, 1)\} \)
- \( R_7 = \{r_7\}, \text{Profile: } \{(e, 1)\} \)
- \( R_8 = \{r_8\}, \text{Profile: } \{(f, 1)\} \)
- \( R_9 = \{r_9\}, \text{Profile: } \{(g, 1)\} \)
- \( R_{12} = \{r_{12}\}, \text{Profile: } \{(h, 1)\} \)
- \( R_{22} = \{r_4, r_3\}, \text{Profile: } \{(d, 1)\} \)
- \( R_{23} = \{r_{11}, r_{10}\}, \text{Profile: } \{(i, 1)\} \)
- \( R_{25} = \{r_{17}, r_{16}, r_{15}, r_{14}, r_{13}, r_2, r_1\}, \text{Profile: } \{(a, 1), (j, 1), (c, 2), (e, 2)\} \)

Assuming that there is no given goal of the inference and \( F = \{(a, 1), (j, 1), (c, 2), (e, 2)\} \) we need to find a group of rules that matches set \( F \).

- \( \text{sim}(F, \text{Profile}(R_5)) = 0 \)
- \( \text{sim}(F, \text{Profile}(R_6)) = 0 \)
- \( \text{sim}(F, \text{Profile}(R_7)) = 0 \)
- \( \text{sim}(F, \text{Profile}(R_8)) = 0 \)
- \( \text{sim}(F, \text{Profile}(R_9)) = 0 \)
- \( \text{sim}(F, \text{Profile}(R_{12})) = 0 \)
- \( \text{sim}(F, \text{Profile}(R_{22})) = 0 \)
- \( \text{sim}(F, \text{Profile}(R_{23})) = 0 \)
- \( \text{sim}(F, \text{Profile}(R_{25})) = 1 \)

If there is more than one relevant group, it is necessary to choose one for further analysis. In our case, there is only one such group. It is group \( R_{25} = \{r_{17}, r_{16}, r_{15}, r_{14}, r_{13}, r_2, r_1\} \).

We need to search for rules to execute within this group: \( \text{sim}(F, \text{Profile}(R_{17})) = 1 \), \( \text{sim}(F, \text{Profile}(R_{16})) = 1 \), \( \text{sim}(F, \text{Profile}(R_{15})) = 1 \), \( \text{sim}(F, \text{Profile}(R_{14})) = 1 \), \( \text{sim}(F, \text{Profile}(R_{13})) = 1 \), \( \text{sim}(F, \text{Profile}(R_2)) = 1 \), \( \text{sim}(F, \text{Profile}(R_1)) = 1 \).

Every rule can be executed because it matches the whole set of facts. It is necessary to use the so-called conflict set strategy[16], which allows for selecting one rule. Using the LAST_RULE strategy, rule \( r_{17} \) is chosen to execute. In effect, conclusion of rule \( r_{17} : (b, 1) \) is added as a new fact to \( F = F \cup \{(b, 1)\} \). Next, according to user preferences, the algorithms runs again until it executes all relevant rules, or a given number of iterations has been done or a given goal of the inference was included in the set of facts.

[15] Due to the limitation of size, the example is very simple and it is directed at presenting a conception described above, useful for improving the classical inference algorithm.

[16] There are many possible conflict set strategies: LAST_RULE, FIRST_RULE, LONGEST_RULE, SHORTEST_RULE
5.2 Results of Experiments

We are aware that experiments should be done on real KBs. The results presented in the article are derived from both real and artificial rules sets. Unfortunately, access to real large rules sets is not easy. It is worth to mention that currently we are obtaining two real KBs with over one and over four thousand of rules. Table 1 presents the results of the experiments carried out on 12 different cases of KBs named 1A...4C. Even if the number of rules is the same for some sets, they differ in sets of facts given as the input knowledge. Thus KBs are considered as different sets. We expect that during the next two months it will be possible to present the results of these experiments. As it can be seen in the table, in case of the classical forward inference process (rule-based) the more rules in the KB, the longer the time of inference. The reason is the necessity of matching every rule with a given set of facts. In contrast, the more rules, the faster partition-based inference. The explanation is simple. The more rules, the more advantages of using the modification of the classical forward inference algorithm in comparison to the classical deduction. The differences in times of inference on two structures of the KB: rules and the partition of rules are getting bigger when the number of rules grows up.

<table>
<thead>
<tr>
<th>KB</th>
<th>rule-based inference</th>
<th>partition of rules-based inference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1A</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>1B</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>1C</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2A</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>2B</td>
<td>17</td>
<td>62</td>
</tr>
<tr>
<td>2C</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>3A</td>
<td>199</td>
<td>30,4</td>
</tr>
<tr>
<td>3B</td>
<td>199</td>
<td>42,9</td>
</tr>
<tr>
<td>3C</td>
<td>199</td>
<td>45,1</td>
</tr>
<tr>
<td>4A</td>
<td>416</td>
<td>51,8</td>
</tr>
<tr>
<td>4B</td>
<td>416</td>
<td>51,8</td>
</tr>
<tr>
<td>4C</td>
<td>416</td>
<td>66,5</td>
</tr>
</tbody>
</table>

The following information is presented in the table: the number of rules (1), the number of rules analyzed during the inference (2)\(^1\), the time of the inference process for rules [ms] (3) and the average time of the inference [ms] (3A), the time of loading the KB for rules (4), the number of groups (5) and the number of groups analyzed during the inference (6), the time of inference process for groups of rules [ms] (7) and the average time of the inference for partition of rules [ms] (7A), the time to build the partition of rules [ms] (8), the time of loading the KB [ms] (9) as well as % of the KB searched in contrast to the classic forward inference process (10) with average % of the KB analyzed in contrast to the classic version (10A).
We have observed that the time of rules partitioning is significantly long in comparison to the time of inference, and the difference increases as the number of rules gets higher. That is why we suggest having the partition of rules stored as a static structure (assuming that it is rebuilt whenever the KB is modified). Being aware of it, we are interested only in reducing the time of the inference maximally.

6 Conclusion

The article presents an idea of a rule-based KB decomposed into groups of rules, called partition of rules. The KB in such a form allows for improving the efficiency of inference algorithms. In this article, only the strategy based on rules clustering is presented. However, there are many possible strategies for partitioning rules. Searching representatives of the partition of rules, instead of every single rule enables reducing the time of inference significantly. The algorithm for the forward inference on the partition of rules (proposed by the authors) has been compared to the classical inference algorithm (for rules). The results presented in Table 1 are promising. We can assume that for large rules sets (consisted of hundreds of rules or more) it is better to partition them into groups of similar rules and to carry out the inference process on representatives of such groups. The partition of rules is a very useful property which makes management of large rules sets easier and speeds up the inference process. The concept of rules’ partitions is a new proposition, based on the previous research concerning rules clusters [9] and decision units [8]. Proposition of the modified inference algorithm is the continuation of the previous research on optimisation of the inference in knowledge-based systems [7]. During next works the authors plan to perform more detailed comparative experiments on real world KBs, consisted of over 4000 rules.

Mining knowledge bases is becoming an important task for many data mining applications. Thus in the future work, to reduce the time complexity of the algorithm for rules clustering and the inference on created groups of rules, the authors are going to present modifications of clustering algorithms based on incomplete knowledge.

Acknowledgements

This work is a part of the project „Exploration of rule knowledge bases” founded by the Polish National Science Centre (NCN: 2011/03/D/ST6/03027).

References


Complexity Studies for Safe and Fan-Bounded Elementary Hornets

Michael Köhler-Bußmeier and Frank Heitmann
michael.koehler-bussmeier@haw-hamburg.de
heitmann@informatik.uni-hamburg.de

Abstract. HORNETS are Petri nets that have nets as tokens. There are an algebraic extension of elementary object nets (EOS) with the possibility to modify the structure of the net-tokens. In previous contributions we investigated elementary HORNETS as well as their subclass of safe elementary HORNETS. We showed that the reachability problem for safe elementary HORNETS requires at least exponential space. We have also showed that exponential space is sufficient. This shows that safe elementary HORNETS are much more complicated than safe elementary object nets (safe EOS), where reachability is known to be PSPACE-complete. In this contribution we study structural restrictions of elementary HORNETS that have a better complexity: fan-bounded HORNETS. It turns out that reachability is again in PSPACE for this class of HORNETS.

Key words: Hornets, nets-within-nets, object nets, reachability, safeness

1 Hornets: Higher-Order Object Nets

In this paper we study self-modifying systems in the formalisms of HORNETS. HORNETS are a generalisation of object nets [1, 2], which follow the nets-within-nets paradigm as proposed by Valk [3].

![Fig. 1. An Elementary Object Net System (EOS)]

With object nets we study Petri nets where the tokens are nets again, i.e. we have a nested marking. Events are also nested. We have three different kinds of events – as illustrated by the example given in Figure 1:

1. System-autonomous: The system net transition \( \hat{t} \) fires autonomously, which moves the net-token from \( \hat{p}_1 \) to \( \hat{p}_2 \) without changing its marking.
2. Object-autonomous: The object net fires transition \( t_1 \) “moving” the black token from \( q_1 \) to \( q_2 \). The object net remains at its location \( \hat{p}_1 \).
3. Synchronisation: Whenever we add matching synchronisation inscriptions at the system net transition $t$ and the object net transition $t_1$, then both must fire synchronously: The object net is moved to $p_2$ and the black token moves from $q_1$ to $q_2$ inside. Whenever synchronisation is specified, autonomous actions are forbidden.

For HORNets we extend object-nets with algebraic concepts that allow to modify the structure of the net-tokens as a result of a firing transition. This is a generalisation of the approach of algebraic nets [4], where algebraic data types replace the anonymous black tokens.

The use of algebraic operations in HORNets relates them to algebraic higher-order (AHO) systems [5], which are restricted to two-levelled systems but have a greater flexibility for the operations on net-tokens, since each net transformation is allowed. There is also a relationship to Nested Nets [6], which are used for adaptive systems.

It is not hard to prove that the general HORNet formalism is Turing-complete. In [7] we have proven that there are several possibilities to simulate counter programs: One could use the nesting to encode counters. Another possibility is to encode counters in the algebraic structure of the net operators.

In our general research we like to study the complexity that arises due the algebraic structure. Therefore, we restrict HORNets to guarantee that the system has a finite state space: First, we allow at most one token on each place, which results in the class of safe HORNets. However this restriction does not guarantee finite state spaces, since we have the nesting depth as a second source of undecidability [2]. Second, we restrict the universe of object nets to finite sets. Finally, we restrict the nesting depth and introduce the class of elementary HORNets, which have a two-levelled nesting structure. This is done in analogy to the class of elementary object net systems (EOS) [1], which are the two-level specialisation of general object nets [1, 2].

If we rule out these sources of complexity the main origin of complexity is the use of algebraic transformations, which are still allowed for safe, elementary HORNets. As a result we obtain the class of safe, elementary HORNets – in analogy to the class of safe EOS [8]. We have shown in [8–10] that most problems for safe EOS are PSPACE-complete. More precisely: All problems that are expressible in LTL or CTL, which includes reachability and liveness, are PSPACE-complete. This means that with respect to these problems safe EOS are no more complex than p/t nets. In a previous publication [11] we have shown that safe, elementary HORNets are beyond PSPACE. We have shown a lower bound, i.e. that “the reachability problem requires exponential space” for safe, elementary HORNets – similarly to well known result of for bounded p/t nets [12]. In [13] we give an algorithm that needs at most exponential space, which shows that lower and upper bound coincide.

In this paper we would like to study restrictions of Elementary HORNets to obtain net classes where the reachability requires less than exponential space. From [11] we know that the main source of complexity for eHORNets is mainly due to the huge number of different of net-tokens, which is double-exponential for safe eHORNets. A closer look reveals that the number of net-token’s marking is rather small – “only” single-exponential, while the number of different object nets is double-exponential. We conclude that restricting the net-tokens’ marking beyond safeness would not improve complexity. Instead, we have to impose structural restrictions on the object-nets. Petri
net theory offers several well known candidates for structural restrictions, like state machines, free-choice nets etc. Here, we restrict object-nets to state-machines.

The paper has the following structure: Section 2 defines Elementary Hornet. Since the reachability problem is known to be undecidable even for EOS, we restrict elementary Hornet to safe ones, which have finite state spaces. State Machines have at most one place in the pre- and in the post-set. In Section 3 we generalise this notion in the way that the number of all places in pre- and postset of an object-net is below a given bound. So, we obtain the maximal synchronisation degree of the objects nets (i.e. the maximal pre- and postset size) as a fresh complexity parameter. Section 4 shows that the reachability problem is PSPACE-complete.

2 Definition of Elementary Hornet (EHornet)

A multiset \( m \) on the set \( D \) is a mapping \( m : D \to \mathbb{N} \). Multisets can also be represented as a formal sum in the form \( m = \sum_{i=1}^{n} x_i \), where \( x_i \in D \).

Multiset addition is defined component-wise: \( (m_1 + m_2)(d) := m_1(d) + m_2(d) \). The empty multiset \( 0 \) is defined as \( 0(d) = 0 \) for all \( d \in D \). Multiset-difference \( m_1 - m_2 \) is defined by \( (m_1 - m_2)(d) := \max(m_1(d) - m_2(d), 0) \).

The cardinality of a multiset is \( |m| := \sum_{d \in D} m(d) \). A multiset \( m \) is finite if \( |m| < \infty \). The set of all finite multisets over the set \( D \) is denoted \( MS(D) \).

Multiset notations are used for sets as well. The meaning will be apparent from its use.

Any mapping \( f : D \to D' \) extends to a multiset-homomorphism \( f^\ast : MS(D) \to MS(D') \) by \( f^\ast(\sum_{i=1}^{n} x_i) = \sum_{i=1}^{n} f(x_i) \).

A p/t net \( N \) is a tuple \( N = (P, T, \text{pre}, \text{post}) \), such that \( P \) is a set of places, \( T \) is a set of transitions, with \( P \cap T = \emptyset \), and \( \text{pre}, \text{post} : T \to MS(P) \) are the pre- and post-condition functions. A marking of \( N \) is a multiset of places: \( m \in MS(P) \). We denote the enabling of \( t \) in marking \( m \) by \( m \downarrow t \). Firing of \( t \) is denoted by \( m \trans{t} m' \).

Net-Algebras We define the algebraic structure of object nets. For a general introduction of algebraic specifications cf. [14].

Let \( K \) be a set of net-types (kinds). A (many-sorted) specification \( (\Sigma, X, E) \) consists of a signature \( \Sigma \), a family of variables \( X = (X_k)_{k \in K} \), and a family of axioms \( E = (E_k)_{k \in K} \).

A signature is a disjoint family \( \Sigma = (\Sigma_{k_1,\ldots,k_n,k})_{k_1,\ldots,k_n,k \in K} \) of operators. The set of terms of type \( k \) over a signature \( \Sigma \) and variables \( X \) is denoted \( \mathcal{T}_{\Sigma}^{X}(X) \).

We use (many-sorted) predicate logic, where the terms are generated by a signature \( \Sigma \) and formulae are defined by a family of predicates \( \Psi = (\Psi_n)_{n \in \mathbb{N}} \). The set of formulae is denoted \( PL_{\Gamma} \), where \( \Gamma = (\Sigma, X, E, \Psi) \) is the logic structure.

Let \( \Sigma \) be a signature over \( K \). A net-algebra assigns to each type \( k \in K \) a set \( \mathcal{U}_k \) of object nets – the net universe. Each object \( N \in \mathcal{U}_k, k \in K \) net is a p/t net \( N = (P_N, T_N, \text{pre}_N, \text{post}_N) \). We identify \( \mathcal{U} \) with \( \bigcup_{k \in K} \mathcal{U}_k \) in the following. We assume the family \( \mathcal{U} = (\mathcal{U}_k)_{k \in K} \) to be disjoint.

The nodes of the object nets in \( \mathcal{U}_k \) are not disjoint, since the firing rule allows to transfer tokens between net tokens within the same set \( \mathcal{U}_k \). Such a transfer is possible,
if we assume that all nets $N \in U_k$ have the same set of places $P_k$. $P_k$ is the place universe for all object nets of kind $k$.

In general, $P_k$ is not finite. Since we like each object net to be finite in some sense, we require that the transitions $T_N$ of each $N \in U_k$ use only a finite subset of $P_k$, i.e. $\forall N \in U : \vert T_N \cup T_N^* \vert < \infty$.

The family of object nets $U$ is the universe of the algebra. A net-algebra $(U, I)$ assigns to each constant $\sigma \in \Sigma_{\lambda, k}$ an object net $\sigma^T \in U_k$ and to each operator $\sigma \in \Sigma_{k_1, \ldots, k_n, k}$ with $n > 0$ a mapping $\sigma^T : (U_{k_1} \times \cdots \times U_{k_n}) \rightarrow U_k$.

A net-algebra is called finite if $P_k$ is a finite set for each $k \in K$.

Since all nets $N \in U_k$ have the same set of places $P_k$, which is finite for eHORNETS, there is an upper bound for the cardinality of $U_k$.

**Proposition 1 (Lemma 2.1 in [11]).** For each $k \in K$ the cardinality of each net universe $U_k$ is bound as follows: $\vert U_k \vert \leq 2^{(\Sigma_{\lambda, k})}$.

A variable assignment $\alpha = (\alpha_k : X_k \rightarrow U_k)_{k \in K}$ maps each variable onto an element of the algebra. For a variable assignment $\alpha$ the evaluation of a term $t \in T^*_k(X)$ is uniquely defined and will be denoted as $\alpha(t)$.

A net-algebra, such that all axioms of $(\Sigma, X, E)$ are valid, is called net-theory.

**Nested Markings** A marking of an eHORNET assigns to each system net place one or many net-tokens. The places of the system net are typed by the function $k : \tilde{P} \rightarrow K$, meaning that a place $\tilde{p}$ contains net-tokens of kind $k(\tilde{p})$. Since the net-tokens are instances of object nets, a marking is a nested multiset of the form:

$$\mu = \sum_{i=1}^{n} \tilde{p}_i [N_i, M_i] \quad \text{where} \quad \tilde{p}_i \in \tilde{P}, N_i \in U_{k(\tilde{p}_i)}, M_i \in MS(P_{N_i}), n \in \mathbb{N}$$

Each addend $\tilde{p}_i [N_i, M_i]$ denotes a net-token on the place $\tilde{p}_i$ that has the structure of the object net $N_i$ and the marking $M_i \in MS(P_{N_i})$. The set of all nested multisets is denoted as $\mathcal{M}_H$. We define the partial order $\sqsubseteq$ on nested multisets by setting $\mu_1 \sqsubseteq \mu_2$ iff $\exists \mu : \mu_2 = \mu_1 + \mu$.

The projection $\Pi^{1,H}_N (\mu)$ is the multiset of all system-net places that contain the object-net $N$:

$$\Pi^{1,H}_N \left( \sum_{i=1}^{n} \tilde{p}_i [N_i, M_i] \right) := \sum_{i=1}^{n} 1_N(N_i) \cdot \tilde{p}_i$$

for $1 \leq i \leq n$, where the indicator function $1_N$ is defined as: $1_N(N_i) = 1$ iff $N_i = N$.

Analogously, the projection $\Pi^{2,H}_N (\mu)$ is the multiset of all net-tokens’ markings (that belong to the object-net $N$):

$$\Pi^{2,H}_N \left( \sum_{i=1}^{n} \tilde{p}_i [N_i, M_i] \right) := \sum_{i=1}^{n} 1_k(N_i) \cdot M_i$$

The projection $\Pi^{2,H}_k (\mu)$ is the sum of all net-tokens’ markings belonging to the same type $k \in K$:

$$\Pi^{2,H}_k (\mu) := \sum_{N \in U_k} \Pi^{2,H}_N (\mu)$$

\footnote{The superscript $H$ indicates that the function is used for HORNETS.}
Synchronisation  The transitions in an HORNET are labelled with synchronisation inscriptions. We assume a fixed set of channels $C = (C_k)_{k \in K}$.

- The function family $\bar{l}_\alpha = (\bar{l}^k_\alpha)_{k \in K}$ defines the synchronisation constraints. Each transition of the system net is labelled with a multiset $\bar{l}^k(\bar{t}) = (e_1, c_1) + \ldots + (e_n, c_n)$, where the expression $e_i \in T^k(X)$ describes the called object net and $c_i \in C_k$ is a channel. The intention is that $\bar{t}$ fires synchronously with a multiset of object net transitions with the same multiset of labels. Each variable assignment $\alpha$ generates the function $\bar{l}^k_\alpha(\bar{t})$ defined as:

$$\bar{l}^k_\alpha(\bar{t})(N) := \sum_{1 \leq i \leq n, \alpha(c_i) \in N} c_i \quad \text{for} \quad \bar{l}^k(\bar{t}) = \sum_{1 \leq i \leq n} (e_i, c_i) \quad (4)$$

Each function $\bar{l}^k_\alpha(\bar{t})$ assigns to each object net $N$ a multiset of channels.

- For each $N \in U_k$ the function $l_N$ assigns to each transition $t \in T_N$ either a channel $c \in C_k$ or $\bot$, whenever $t$ fires without synchronisation, i.e. autonomously.

System Net  Assume we have a fixed logic $\Gamma = (\Sigma, X, E, \Psi)$ and a net-theory $(U, I)$. An elementary higher-order object net (eHORNET) is composed of a system net $\hat{N}$ and the set of object nets $U$. W.l.o.g. we assume $\hat{N} \not\in U$. To guarantee finite algebras for eHORNets, we require that the net-theory $(U, I)$ is finite, i.e. each place universe $P_k$ is finite.

The system net is a net $\hat{N} = (\hat{P}, \hat{T}, \text{pre}, \text{post}, \hat{G})$, where each arc is labelled with a multiset of terms: $\text{pre}, \text{post} : \hat{T} \rightarrow (\hat{P} \rightarrow MS(T^k(X)))$. Each transition is labelled by a guard predicate $\hat{G} : \hat{T} \rightarrow PL_I$. The places of the system net are typed by the function $k : \hat{P} \rightarrow K$. As a typing constraint we have that each arc inscription has to be a multiset of terms that are all of the kind that is assigned to the arc’s place:

$$\text{pre}(\bar{t})(\bar{p}), \quad \text{post}(\bar{t})(\bar{p}) \in MS(T^k(X)) \quad (5)$$

For each variable binding $\alpha$ we obtain the evaluated functions $\text{pre}_\alpha, \text{post}_\alpha : \hat{T} \rightarrow (\hat{P} \rightarrow MS(U))$ in the obvious way.

Definition 1  (Elementary Hornet, eHORNET). Assume a fixed many-sorted predicate logic $\Gamma = (\Sigma, X, E, \Psi)$.

An elementary HORNET is a tuple $EH = (\hat{N}, U, I, k, l, \mu_0)$ such that:

1. $\hat{N}$ is an algebraic net, called the system net.
2. $(U, I)$ is a finite net-theory for the logic $\Gamma$.
3. $k : \hat{P} \rightarrow K$ is the typing of the system net places.
4. $l = (l_N)_{N \in U}$ is the labelling.
5. $\mu_0 \in M_H$ is the initial marking.

Events  The synchronisation labelling generates the set of system events $\Theta$. We have three kinds of events:
1. Synchronised firing: There is at least one object net that has to be synchronised, i.e. there is a $N$ such that $\hat{t}(\delta)(N)$ is not empty.

   Such an event is a pair $\theta = \hat{t}[\delta]$, where $\hat{t}$ is a system net transition, $\alpha$ is a variable binding, and $\delta$ is a function that maps each object net to a multiset of its transitions, i.e. $\delta(N) \in MS(T_N)$. It is required that $\hat{t}$ and $\delta(N)$ have matching multisets of labels, i.e. $\hat{t}(\delta)(N) = \ell^2_N(\delta(N))$ for all $N \in \mathcal{U}$. (Remember that $\ell^2_N$ denotes the multiset extension of $l_N$.)

   The intended meaning is that $\hat{t}$ fires synchronously with all the object net transitions $\delta(N)$, $N \in \mathcal{U}$.

2. System-autonomous firing: The transition $\hat{t}$ of the system net fires autonomously, whenever $\hat{t}(\delta)$ is the empty multiset $0$.

   We consider system-autonomous firing as a special case of synchronised firing generated by the function $\delta_{id}$, defined as $\delta_{id}(N) = 0$ for all $N \in \mathcal{U}$.

3. Object-autonomous firing: An object net transition $t$ in $N$ fires autonomously, whenever $l_N(t) = \bot_k$.

   Object-autonomous events are denoted as $id_{\hat{p},N}[\theta_1]$, where $\delta_1(N') = \{t\}$ if $N = N'$ and $0$ otherwise. The meaning is that in object net $N$ fires $t$ autonomously within the place $\hat{p}$.

   For the sake of uniformity we define for an arbitrary binding $\alpha$:

   $\text{pre}_\alpha(id_{\hat{p},N}[\delta_1])(\hat{p}')(N') = \text{post}_\alpha(id_{\hat{p},N}[\delta_1])(\hat{p}')(N') = \begin{cases} 1 & \text{if } \hat{p}' = \hat{p} \land N' = N \\ 0 & \text{otherwise.} \end{cases}$

   The set of all events generated by the labelling $l$ is $\Theta_l := \Theta_1 \cup \Theta_2$, where $\Theta_1$ contains synchronous events (including system-autonomous events as a special case) and $\Theta_2$ contains the object-autonomous events:

   $\Theta_1 := \left\{ \hat{\tau}_\alpha[\delta] \mid \forall N \in \mathcal{U} : \hat{\tau}_\alpha(\hat{t})(N) = \ell^2_N(\delta(N)) \right\}$

   $\Theta_2 := \left\{ id_{\hat{p},N}[\delta_1] \mid \hat{p} \in \hat{P}, N \in \mathcal{U}_{\hat{p}}, t \in T_N \right\}$

(6)

**Firing Rule** A system event $\theta = \hat{\tau}_\alpha[\delta]$ removes net-tokens together with their individual internal markings. Firing the event replaces a nested multiset $\lambda \in \mathcal{M}_H$ that is part of the current marking $\mu$, i.e. $\lambda \subseteq \mu$, by the nested multiset $\rho$. The enabling condition is expressed by the enabling predicate $\phi_{EH}$ (or just $\phi$ whenever $EH$ is clear from the context):

$\phi_{EH}(\hat{\tau}_\alpha[\delta], \lambda, \rho) \leftrightarrow \forall k \in K :$

$\forall \hat{p} \in k^{-1}(k) : \forall N \in \mathcal{U}_k : \Pi^1_H(\lambda)(\hat{p}) = \text{pre}_\alpha(\hat{\tau})(\hat{p})(N) \land$

$\forall \hat{p} \in k^{-1}(k) : \forall N \in \mathcal{U}_k : \Pi^1_H(\rho)(\hat{p}) = \text{post}_\alpha(\hat{\tau})(\hat{p})(N) \land$

$\Pi^2_H(\lambda) \geq \sum_{N \in \mathcal{U}_k} \pi^1_N(\delta(N)) \land$

$\Pi^2_H(\rho) = \Pi^2_H(\lambda) + \sum_{N \in \mathcal{U}_k} \pi^2_N(\delta(N)) - \text{pre}_\lambda^2_N(\delta(N))$ (7)

The predicate $\phi_{EH}$ has the following meaning: Conjunct (1) states that the removed sub-marking $\lambda$ contains on $\hat{p}$ the right number of net-tokens, that are removed by $\hat{\tau}$. 
Conjunct (2) states that generated sub-marking \( \rho \) contains on \( \tilde{p} \) the right number of net-tokens, that are generated by \( \tilde{\tau} \). Conjunct (3) states that the sub-marking \( \lambda \) enables all synchronised transitions \( \vartheta(N) \) in the object \( N \). Conjunct (4) states that the marking of each object net \( N \) is changed according to the firing of the synchronised transitions \( \vartheta(N) \).

Note, that conjunct (1) and (2) assures that only net-tokens relevant for the firing are included in \( \lambda \) and \( \rho \). Conditions (3) and (4) allow for additional tokens in the net-tokens.

For system-autonomous events \( \tilde{\tau}^{\sigma}[\vartheta_{\text{id}}] \) the enabling predicate \( \phi_{E \text{H}} \) can be simplified further: Conjunct (3) is always true since \( \text{pre}_{N}(\vartheta_{\text{id}}(N)) = 0 \). Conjunct (4) simplifies to \( \Pi_{k}^{1,H}(\rho) = \Pi_{k}^{2,H}(\lambda) \), which means that no token of the object nets get lost when a system-autonomous events fires.

Analogously, for an object-autonomous event \( \tilde{\tau}[\vartheta_{t}] \) we have an idle-transition \( \tilde{\tau} = id_{\tilde{p}},N \) and \( \vartheta = \vartheta_{t} \) for some \( t \). Conjunct (1) and (2) simplify to \( \Pi_{N'}^{1,H}(\lambda) = \tilde{\rho} = \Pi_{N'}^{1,H}(\rho) \) for \( N' = N \) and to \( \Pi_{N'}^{1,H}(\lambda) = 0 = \Pi_{N'}^{1,H}(\rho) \) otherwise. This means that \( \lambda = \tilde{p}[M], M \) enables \( t \), and \( \rho = \tilde{p}[M - \text{pre}_{N}(\tilde{t}) + \text{post}_{N}(\tilde{t})] \).

**Definition 2 (Firing Rule).** Let \( E \text{H} \) be an \( \text{EHornet} \) and \( \mu, \mu' \in \mathcal{M}_{H} \) markings.

- The event \( \tilde{\tau}^{\sigma}[\vartheta] \) is enabled in \( \mu \) for the mode \( (\lambda, \rho) \in \mathcal{M}_{H}^{2} \) iff \( \vartheta \subseteq \mu \land \phi_{E \text{H}}(\tilde{\tau}[\vartheta], \lambda, \rho) \) holds and the guard \( G(\tilde{t}) \) holds, i.e. \( E \models_{\sigma} G(\tilde{\tau}) \).

- An event \( \tilde{\tau}^{\sigma}[\vartheta] \) that is enabled in \( \mu \) can fire − denoted \( \mu \xrightarrow{E \text{H}}_{\tilde{\tau}[\vartheta](\lambda, \rho)} \mu' \).

- The resulting successor marking is defined as \( \mu' = \mu - \lambda + \rho \).

Note, that the firing rule has no-apriori decision how to distribute the marking on the generated net-tokens. Therefore we need the mode \( (\lambda, \rho) \) to formulate the firing of \( \tilde{\tau}^{\sigma}[\vartheta] \) in a functional way.

### 3 Fan-Bounded Safe, Elementary HORNETS

We know from [11, Lemma 3.1] that a safe \( \text{EHornet} \) has a finite reachability set. More precisely: There are at most \( \left(1 + U(m) \cdot 2^{m}\right)^{|P|} \) different markings, where \( m \) is the maximum of all \( |P_{k}| \) and \( U(m) \) is the number of object nets. In the general case we have \( U(m) = 2^{2^{km}} \), which dominates the bound. It is double-exponential, while the number of different marking within each net-token is \( 2^{m} \), i.e. “only” single-exponential. The huge number of object nets is the source of the exponential space requirement for the reachability problem.

Therefore, if one wants to require less than exponential space one has to restrict the structure of possible object nets in \( \mathcal{U}_{k} \). The huge number of object-nets in \( \mathcal{U}_{k} \) arises since we allow object nets with any number of places in the preset or postset, i.e. unbounded joins or forks. Therefore it seems promising to restrict the synchronisation degree.

In the following we want to restrict the number of object-nets in \( \mathcal{U}_{k} \). We forbid unbounded joins or forks. From a practical point of view this can be considered as a rather unlikely. From a theoretical point of view we can take this into account with an parametrised complexity analysis. The parameter considered here is the maximal
number of places in the pre- or postset, i.e. the maximal synchronisation at object-net level.

**Definition 3.** An elementary HORNET \( EH = (\hat{N}, \mathcal{U}, \mathcal{I}, \lambda, l, \mu_0) \) is called \( \beta \)-fan-bounded whenever all transitions of all object-nets have at most \( \beta \) places in the pre- and in the postset:

\[ \forall k \in K : \forall N \in \mathcal{U}_k : \forall t \in T_N : |\star t| \leq \beta \land |t^\bullet| \leq \beta \]

The fan-bound of \( EH \) is defined as:

\[ \beta(EH) := \max \{ |\star t|, |t^\bullet| : k \in K : N \in \mathcal{U}_k : t \in T_N \} \]

Note, that an elementary HORNET is always fan-bounded, since \( P_N \subseteq P_k \) and \( P_k \) is always finite in the elementary case: \( \beta(EH) \leq |P_k| < \infty \).

**Proposition 2.** For a safe, \( \beta \)-fan-bounded eHORNET the cardinality of each net universe \( \mathcal{U}_k \) is bounded for each \( k \in K \) as follows: \( |\mathcal{U}_k| \leq 2^{O(n^{(4\beta)})} \) where \( n := |P_k| \).

**Proof.** For a safe, \( \beta \)-fan-bounded eHORNET the number of possible objects is calculated as follows: Each possible transition \( t \) chooses a subset of \( P_k \) for the preset \( \star t \) and another subset for the postset \( t^\bullet \) with the constraint that these subsets have a cardinality of at most \( \beta \). The number of these subsets is:

\[ \left| \bigcup_{i=0}^{\beta} \binom{P_k}{i} \right| = \sum_{i=0}^{\beta} \binom{|P_k|}{i} = \binom{|P_k|}{0} + \binom{|P_k|}{1} + \cdots + \binom{|P_k|}{\beta} \]

(Here \( \binom{A}{i} \) denote the set of all subsets of \( A \) that have cardinality \( i \).)

We identify \( t \) with the pair \((\star t, t^\bullet)\). The number of different transitions is:

\[ |T_k| = \left( \binom{|P_k|}{0} + \binom{|P_k|}{1} + \cdots + \binom{|P_k|}{\beta} \right)^2 \]

\[ \leq (1 + n + \frac{n(n-1)}{2!} + \cdots + \frac{n(n-1)\cdots(2)}{\beta!})^2 \]

\[ \leq \left( \text{const} \cdot n^\beta \right)^2 \]

\[ = \text{const} \cdot n^{2\beta} \]

So, the number of different transitions is in \( O(n^{2\beta}) \).

The set of labelled transitions is \( LT_k := T_k \times (C_k \cup \{\perp_k\}) \) and we have \( |LT_k| = |T_k| \cdot |C_k \cup \{\perp_k\}| \) different labelled transitions. We cannot use more channels than we have transitions in the object net, i.e. we could use at most \( |T_k| \) different channels from \( C_k \cup \{\perp_k\} \). Thus, we have:

\[ |LT_k| = |T_k| \cdot (|C_k| + 1) \leq |T_k| \cdot |T_k| \]

Note, that while the bound we have given for the general case in Lemma 2.1 in [11] is strict (i.e. there are Hornets that exactly have this number of object-nets) the calculation given here gives us only an upper bound.
From $|T_k| \leq const \cdot n^{(2\beta)}$ we obtain:

$$|LT_k| \leq (const \cdot n^{2\beta})^2 = const \cdot n^{(4\beta)}$$

Thus the set of labelled transitions is in $O(n^{(4\beta)})$, i.e. a polynomial in the number of places $n = |P_k|$ where the degree of the polynomial is given by the fan-parameter $\beta$.

Since each object net $N$ in $U_k$ is characterised by its set of labelled transitions and there are $|\mathcal{P}(LT_k)| = 2^{|LT_k|}$ subsets of $LT_k$, we have at most $2^{O(n^{(4\beta)})}$ different object nets.

Thus the set of different object nets is only single-exponential for fan-bounded EHOR-NETS – and not double-exponential as in the general case.

Note, that the set of transitions is a polynomial in the number of places $m$ where the degree is given by the fan-parameter $\beta = \beta(m) \leq m$. Of course if we have transitions that use all the places in the pre- or postset, i.e. $\beta = m$ we have an exponential number as before, since:

$$|T_k| = \left( \binom{|P_k|}{0} + \binom{|P_k|}{1} + \cdots + \binom{|P_k|}{m} \right)^2 = \left( 2^{|P_k|} \right)^2 = 2^{(2|P_k|)}$$

So, the general analysis is just the special case where the fan-parameter $\beta$ equals the number of places $m$.

For safe, $\beta$-fan-bounded EHORNET we can give an upper bound for the number of reachable markings. The number of reachable markings is in $2^{O(n^{(4\beta+1)})}$, i.e. exponential, where the exponent is a polynomial in the number of places $n$ where the degree is given by the fan-parameter $\beta$.

**Proposition 3.** A safe, $\beta$-fan-bounded EHORNET has a finite reachability set.

The number of reachable markings is bounded by $2^{O(n^{(4\beta+1)})}$ where $n$ is the maximum of all $|P_k|$ and $|\hat{P}|$.

**Proof.** Analogously Prop. 1 we have at most $(1 + U(m) \cdot 2^m)^{|\hat{P}|}$ different markings in the safe HORNET.

For a $\beta$-fan-bounded EHORNET we have obtained in Prop. 2 a bound for the number of possible object-nets: $|U_k| \leq U(m) = 2^{(const \cdot m^{(4\beta)})}$. Thus the number of different markings in the safe, $\beta$-fan-bounded EHORNET is:

$$(1 + U(m) \cdot 2^m)^{|\hat{P}|} \leq \left( 1 + 2^{(const \cdot m^{(4\beta)})} \cdot 2^m \right)^{|\hat{P}|} \leq 2^{\left( 2^{(const \cdot m^{(4\beta)})} + m \right)^{|\hat{P}|}} \leq 2^{\left( 2^{(const \cdot m^{(4\beta)})} \right)^{|\hat{P}|}} = 2^{\left( const \cdot m^{(4\beta)} \right)^{|\hat{P}|}}$$

With $n := max(m, |\hat{P}|)$ the bound simplifies to:

$$2^{\left( const \cdot m^{(4\beta)} \right)^{|\hat{P}|}} \leq 2^{\left( const \cdot n^{(4\beta)} \cdot n \right)} = 2^{\left( const \cdot n^{(4\beta+1)} \right)}$$
The number of reachable markings is in $2^{O(n^{4\beta+1})}$, i.e. exponential, where the exponent is a polynomial.

The most extreme restriction is to forbid forks and joins at all. In this case we consider elementary HORNets that have state-machines as object-nets only, i.e.

$$\forall k \in K : \forall N \in U_k : \forall t \in T_N : |t| \leq 1 \land |t^*| \leq 1$$

An elementary Hornet with this restriction is called eSMHORNET (elementary state-machine HORNET) for short. An eSMHORNET is 1-fan-bounded eHORNET by definition. Therefore, we obtain the following corollary of Lemma 3:

**Corollary 1.** A safe eSMHORNET has a finite reachability set. The number of reachable markings is bounded by $2^{O(n^5)}$ where $n$ is the maximum of all $|P_k|$ and $|\hat{P}|$.

### 4 Complexity of the Reachability Problem

As p/t nets are a special subcase of fan-bounded eHORNETs (we simply restrict the system net to a single place with the p/t net of interest as the unique net-token) the reachability problem for safe, fan-bounded eHORNETs cannot be simpler than for p/t nets, i.e. it is at least PSPACE-hard, since the reachability problem for safe p/t nets is PSPACE-complete. In the following we show that the reachability problem for lies within PSPACE.

**Lemma 1.** For safe, $\beta$-fan-bounded eHORNETs there exists a non-deterministic algorithm that decides the reachability problem within polynomial space:

$${\text{Reach}}_{\beta\text{-seH}} \in N\text{Space} \left(O \left(n^{(4\beta+1)} \right) \right)$$

where $n$ is the maximum of all $|P_k|$ and $|\hat{P}|$.

**Proof.** Whenever $\mu^*$ is reachable it is reachable by a firing sequence without loops. The main idea of the algorithm is to guess a firing sequence $\mu_0 \overset{\theta_1}{\rightarrow} \mu_1 \overset{\theta_2}{\rightarrow} \cdots \overset{\theta_{\max}}{\rightarrow} \mu_{\max} = \mu^*$, where $\mu^*$ is the marking to be tested for reachability.

By Prop. 3 we know we have at most $\max = 2^{O(n^{(4\beta+1)})}$ different markings. Therefore, we can safely cut off the computation after $\max$ steps.

For each step $\mu_i \overset{\theta}{\rightarrow} \mu_{i+1}$ we choose non-deterministically some event $\theta_i$. For a given marking $\mu_i$ we guess an event $\theta_i$ and a marking $\mu_{i+1}$ and test whether $\mu_i \overset{\theta_i}{\rightarrow} \mu_{i+1}$ holds.

- The markings $\mu_i$ and $\mu_{i+1}$ can be stored in $O \left(n^{(4\beta+1)} \right)$ bits, i.e. polynomial space.
- The event $\theta_i$ can be stored in polynomial bits: The choice for the system net transition $\hat{t}$ fits in polynomial space. The variable binding $\alpha$ selects for each variable in the arc inscriptions some object net from the universe $U_k$. Since we have always have a finite number of variables polynomial space is sufficient. For each kind $k$ we have the multiset of channels $\hat{P}_k(\hat{t})$ to synchronise with. In the proof of Prop 2 we have seen that we have at most $\text{const} \cdot n^{(4\beta)}$ labelled transitions in the object nets, i.e. polynomial space is sufficient. We guess a mode $(\lambda, \rho)$. (Since we consider safe HORNets the choice for $\lambda$ is unique, since there is at most one net-token on each
place, which implies that whenever an event $\tilde{t}^\alpha[\rho]$ is enabled, then $\lambda$ is uniquely determined. For the multiset $\rho$, we use the fact that for each place in the object net $N$ we have at most one token to distribute over all generated net-tokes. For each net $N \in U_k$ we select one of the net-tokens generated in the postset. All these choices need at most polynomial many bits.

- To check whether the event is enabled we have to whether test $\alpha \sqsubseteq \mu$. This holds iff $\exists \mu'' : \mu = \alpha + \mu''$. Since $\alpha$ and $\mu$ are known, this can be tested in-place by ‘tagging’ all addends from $\alpha$ in $\mu$.

Finally we check whether the successor marking $\mu' = \mu - \lambda + \rho$ is equal to $\mu_{i+1}$. This can be done in-place as $\mu - \lambda$ are those addends that have not been tagged.

After each step $\mu_i \xrightarrow{\theta_i} \mu_{i+1}$ we decrement a counter (which has been initialised with the maximal sequence length $\max$), forget $\mu_i$, and repeat the procedure with $\mu_{i+1}$ again until either the marking of interested is reached or the counter reaches zero.

As each step can be tested in polynomial space, the whole algorithm needs at most polynomial many bits to decide reachability.

Now we use the technique of Savitch to construct a deterministic algorithm from the non-deterministic algorithm above.

**Proposition 4.** The reachability problem $\text{Reach}_{\beta\text{-seH}}$ for safe $\beta$-fan-bounded $\text{E}H\text{ORNETS}$ can be solved within polynomial space.

**Proof.** We known by Lemma 1 that $\text{Reach}_{\beta\text{-seH}} \in \text{NSpace}(O(n^{4\beta+1}))$. From Savitch’s Theorem we obtain:

$$\text{Reach}_{\beta\text{-seH}} \in \text{DSpace}(O(n^{4\beta+1} \cdot 2) = \text{DSpace}(O(n^{2(4\beta+1)}))$$

This proves our central result that the reachability problem for safe, fan-bounded $\text{E}H\text{ORNETS}$ is $\text{PSPACE}$-complete, while it requires exponential space in the general case of safe $\text{E}H\text{ORNETS}$.

**References**

Monitoring with Parametrized Extended Life Sequence Charts

Ming Chai¹ and Bernd-Holger Schlingloff²

¹ ² Humboldt Universität zu Berlin
² Fraunhofer FOKUS
(ming.chai,hs)@informatik.hu-berlin.de

Abstract. Runtime verification is a lightweight verification technique that checks whether an execution of a system satisfies a given property. A problem in monitoring specification languages is to express parametric properties, where the correctness of a property depends on both the temporal relations of events, and the data carried by events. In this paper, we introduce parametrized extended live sequence charts (PeLSCs) for monitoring sequences of data-carrying events. The language of PeLSCs is extended from life sequence charts by introducing condition and assignment structures. We develop a translation from PeLSCs into the hybrid logic $\mathcal{HL}$, and prove that the word problem of the PeLSCs is linear with respect to the size of a parametrized event trace. Therefore, the formalism is feasible for on-line monitoring.

1 Introduction

Even with most advanced quality assurance techniques, correctness of complex software can never be guaranteed. To solve this problem, runtime verification has been proposed to provide on-going protection during the operational phase. Runtime Verification checks whether an execution of a computational system satisfies or violates a given correctness property. It is performed by using a monitor. This is a device or a piece of software that observes the system under monitoring (SuM) and generates a certain verdict (true or false) as the result. Compared to model checking and testing, this technique is considered to be a lightweight validation technique, since it does not try to cover all possible executions of the SuM. It detects failures of an SuM directly in its actual running environment. This avoids some problems of other techniques, such as imprecision of the model in model checking, and inadequateness of the artificial environment in testing.

An execution of a computational system checked by a monitor can be formalized by a sequence of events. One of the challenges in building a runtime verification system is to define a suitable specification language for monitoring properties. A monitoring specification language should be expressive and attractive [22]: The language should be able to express all expected monitoring properties, and the language should keep the formulations simple for simple properties. A simple formulation means that the size of

* This work was supported by the State Key Laboratory of Rail Traffic Control and Safety (Contract No.: RCS2012K001), Beijing Jiaotong University
the formulation is small, and the notations of the formulation is understood by users (e.g., system designers).

Over the last years, various runtime verification systems have been developed which use some form of temporal logic, including linear temporal logic (LTL), metric temporal logic (MTL), time propositional temporal logic (TPTL) and first-order temporal logic (LTLFO). Although these specification languages are expressive and technically sound for monitoring, software engineers are not familiar with them and need extensive training to use them efficiently. Therefore, many runtime verification systems support also other specification languages, such as regular expressions and context-free grammars. Unfortunately, it is difficult to specify properties for parallel systems in these languages, and they are not (yet) used in practice by system designers.

In previous work [14], we proposed an extension of live sequence chart (LSC) [18] for expressing monitoring properties. LSC is a visual formalism that specifies the temporal relations of the exchange of messages among instances. It extends the classical message sequence chart formalism (MSC) by introducing possible and mandatory elements, including universal and existential charts, and hot and cold messages and conditions. With these extensions, LSCs are able to distinguish between required and allowed behaviours of an SuM. Our language of the proposed extended LSCs (eLSCs) introduces modal pre-charts. That is, we distinguish between pre-charts that are necessary conditions of main-charts and those that are sufficient conditions of main-charts.

The eLSC-based monitoring approach so far can not handle parametric properties, where the correctness of a property depends on both the temporal relations of events and data carried by the events. One possible workaround for this shortage is to formalize each assignment of data with a unique atomic proposition. However, since the domain of data can be infinite or unknown, this approach is not sufficient in general. We extend eLSC to parametrized eLSC (PeLSC) by introducing assignment structures and condition structures.

In this paper, we model data-carrying events with parametrized events, where the data is represented by parameters. Consider a client/server system that allows clients to access a server, and consider the following properties.

\[(P1): \text{If there is a log in to the server, it must be followed by a log out.}\]

\[(P2): \text{A log out event can not occur, unless it is preceded by a log in.}\]

\[(P3): \text{If a client logs in to the server, it must log out within 200 sec.}\]

For the first two properties, the monitor can observe a propositional events login and logout. The expected behaviours can be formalized by the following regular expressions:

\[L_1 \triangleq \Sigma^* \circ \{\text{login}\} \circ \{\text{logout}\} \circ \Sigma^*\]

\[L_2 \triangleq \Sigma^* \circ \{\text{login}^+\} \circ \{\text{logout}\} \circ \Sigma^*\]

For the property \((P3)\), each of the login and logout events carries a client name and a time stamp. An execution of this system can be formalized by a sequence of
parametrized events. Each of the propositional events carries two parameters client_id (id) and time_stamp (time). With these definitions, property (P3) can be written more formally as follows:

When the system emits a login event with \((id = x)\) and \((time = y)\), a logout event with \((id = x')\) and \((time = y')\) should occur afterwards, where \((x' = x)\) and \((y' \leq (y + 200))\).

The PeLSCs of Fig. 1 specify the three properties above. The chart (C1) is a standard LSC formalizing (P1). Property (P2) cannot be formalized with LSCs; an eLSC for it is (C2). Property (P3) involves parameters on infinite or unknown domains and thus cannot be expressed by eLSCs. A PeLSC for it is (PU). Formal definitions being given below, we note that the language PeLSC contains variables, assignment and conditions for dealing with data-parametrized events. An assignment structure is used to store an arbitrary parameter value, and a condition structure is used to express constraints on such values. With these extensions, PeLSCs can be used for monitoring systems where events carry data.

To generate monitors, we translate PeLSCs into (a subclass of) the hybrid logic (HL) [13]. HL has a type of symbols called nominals that represent names of parameters. Let \(s\) be a symbol, an HL formula may contain the downarrow binder “\(x \downarrow s\)”. When evaluating an HL formula over a parametrized event trace, the downarrow binder assigns all variables \(x\) in the formula to the value of the parameter \(s\) of the “current” parametrized event.

Fig. 1. Examples: PeLSCs for properties of a client/server system

A monitor essentially solves the word-problem: given a trace, decide whether the trace is in the language defined by a monitoring property. As a main result of this paper, we prove that the complexity of the word-problem of PeLSCs is linear if the propositions in the condition structures express only comparisons of parameter values. Thus, monitoring can be done on-line, while the SuM is running.

The rest of the paper is organized as follows. Section 2 outlines related work. Section 3 introduces parametrized eLSCs (PeLSCs), including their syntax and trace-based semantics. Section 4 presents a translation from PeLSCs into a subclass of HL, and
proves the complexity of the word problem of PeLSCs. Section 5 contains some conclusions and hints for future work.

2 Related Work

Our extension of LSCs is inspired by the treatment of time in live sequence charts proposed by Harel et. al. [21]. There, a time constraint in LSCs is defined by a combination of assignment structures and condition structures. In contrast, we provide a more general notation for arbitrary data parameters.

There are several other runtime verification approaches for handling parametrized events. The EAGLE logic [5], which is a linear $\mu$-calculus, is one of the first logics in runtime verification for specifying and monitoring data-relevant properties. Although EAGLE has rich expressiveness, it has high computational costs [7]. To avoid this problem, other rule-based methods have been introduced. They are based on MetateM [4] and the Rete algorithm [19]. MetateM provides a framework of executing temporal formulae and Rete is an efficient algorithm for matching patterns with objects. Inspired by MetateM, RuleR is an efficient rule-based monitoring system that can compile various temporal logics [7]. LogicFire is an internal domain specification language for artificial intelligence on basis of Rete [23]. The rule-based runtime verification systems have high performance. However, their implementations are still complex. The language of PeLSCs has a comparable expressiveness. However, the implementation of PeLSC based runtime verification system is easier because monitors are generated automatically with the translation algorithm.

TraceMatches [1] is essentially a regular expression language. It extends the language of AspectJ [24] by introducing free variables in the matching patterns. TraceContract is an API for trace analysis, implemented in Scala, which is able to express parametric properties with temporal logic [6]. Monitoring oriented programming (MOP) is an efficient and generic monitoring framework that integrates various specification languages [16]. In particular, JavaMOP deals with parametric specification and monitoring using TraceMatches [25]. TraceMatches and JavaMOP are defined on the basis of trace slicing, which translates parametrized events into propositional events. With trace slicing, the problem of checking parametrized event traces is translated into a (standard) propositional word problem. Although JavaMOP has high performance, to our opinion its expressiveness is insufficient. As pointed out in [3], trace slicing can only handle traces where all events with the same name carry the same parameters. Our PeLSCs based approach overcomes this shortage by using formula rewriting algorithms.

Another important direction of parametric monitoring is based on automata theory. Quantified event automata [3] are an extension of the trace slicing methods mentioned above. They are strictly more expressive than TraceMatches. In that context, data automata have been proposed as a down-scaled version of Rete to an automaton-based formalism [22]. Unfortunately, properties of parallel systems have complex formulations when expressed by automata. PeLSCs can keep the monitoring specification attractive when dealing with such properties.

Various extensions of LTL have been proposed for parametric monitoring. If time is the only parameter, properties can be formalized with real-time logics such as TLTL
For other parameters, first order extensions of LTL have been introduced. Parametrized LTL [28] contains a binary binding operator, and is further translated into parametrized automata for monitoring. First-order temporal logic \( LTL^{FO} \) includes both first-order and temporal connectives [26]. For monitoring \( LTL^{FO} \) an algorithm using a spawning automaton has been developed [11]. However, the word problem of \( LTL^{FO} \) is PSPACE-complete [11], and the translation has a potential of suffering from the state explosion problem. A domain-specific language for monitoring the exchange of XML messages of web service is \( LTL^{FO} \) [20]. This language has a lower complexity than full first order temporal logic. However, its expressiveness is limited by only allowing to express equivalence of variables. Metric Temporal First-order Logic (MFOTL) adds quantifiers to MTL [9], and has been used for monitoring data applications [8].

A domain-specific language for monitoring the exchange of XML messages of web service is \( LTL^{FO} \) [20]. This language has a lower complexity than full first order temporal logic. However, its expressiveness is limited by only allowing to express equivalence of variables. Metric Temporal First-order Logic (MFOTL) adds quantifiers to MTL [9], and has been used for monitoring data applications [8]. An MFOTL monitoring system has been built based on a trace decomposing technique, which may introduce additional errors/mistakes. Similiar to the languages of automata, all these temporal logics have difficulties in specifying concurrency properties. The language of PeLSCs can avoid these shortcomings. The word problem for PeLSCs is linear with respect to the size of traces. Meanwhile, PeLSCs have richer expressiveness than \( LTL^{FO} \) by allowing to express general comparisons of terms.

Our rewriting based algorithms avoid the problems introduced by the LTL to automata translations, and the trace decomposing techniques.

### 3 Definitions of Parametrized eLSCs

This section presents the syntax and semantics of parametrized extended LSCs (PeLSCs). The PeLSCs are interpreted over parametrized event traces, which are defined as follows.

Let \( \Sigma \triangleq \{e_1, e_2, ..., e_n\} \) be a finite alphabet of events, \( N \triangleq \{s_1, s_2, ...\} \) be a countable set of nominals and \( D \triangleq \{d_1, d_2, ...\} \) any domain (e.g., integers, strings, or reals). A parameter is a pair \( p \triangleq (s, d) \) from \( N \times D \), where \( s \) is the name of \( p \) and \( d \) is the value of \( p \).

**Definition 1 (Parametrized event).** Given an alphabet \( \Sigma \) of events, a set \( N \) of nominals and a domain \( D \), a parametrized event is a pair \( w \triangleq (e, \mathcal{P}) \), where \( e \in \Sigma \) is an event and \( \mathcal{P} \in 2^{N \times D} \) is a set of parameters.

Given a parametrized event \( w \) with \( \mathcal{P} \triangleq \{(s_1, d_1), ..., (s_m, d_m)\} \), we define \( \text{Evet}(w) \triangleq e, \text{Para}(w) \triangleq \mathcal{P} \) and \( \text{Nam}(w) \triangleq \{s_1, ..., s_m\} \). A parametrized event \( (e, \mathcal{P}) \) is deterministic if each parameter name in \( \mathcal{P} \) is unique, i.e., for all \( p, p' \in \mathcal{P} \) it holds that \( s \neq s' \). In this paper, we assume that all parametrized events are deterministic.

Parametrized event traces basically are finite sequences of parameterized events.

**Definition 2 (Parametrized event trace).** Given \( N \) and \( D \), a parameter trace \( \rho \triangleq (\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n) \) over \( N \times D \) is a finite sequence of sets of parameters, i.e., an element of \( (2^{N \times D})^* \). Given \( \Sigma, N \) and \( D \), a parametrized event trace \( \tau \triangleq (\sigma, \rho) \) is a pair of a finite event trace \( \sigma \) and a parameter trace \( \rho \) with the same length, i.e., \( \sigma \in \Sigma^* \) and \( \rho \in (2^{N \times D})^* \) and \( |\sigma| = |\rho| \).

By \( \tau[i] \triangleq (\sigma[i], \rho[i]) \) we denote the \( i^{th} \) parametrized event of \( \tau \), where \( \sigma[i] \) and \( \rho[i] \) are the \( i^{th} \) element of \( \sigma \) and \( \rho \), respectively.
3.1 Syntax of PeLSCs

A universal PeLSC consists of two basic charts: a pre-chart and a main-chart. A basic chart is visually similar to an MSC. It specifies the exchange of messages among a set of instances. Each instance is represented by a lifeline. Lifelines in a basic chart are usually drawn as vertical dashed lines, and messages are solid arrows between lifelines. For each message, there are two actions: the action of sending the message and the action of receiving it. Each action occurs at a unique position in a lifeline. The partial order of actions induced by a basic chart is as follows.

- An action at a higher position in a lifeline precedes an action at a lower position in the same lifeline; and
- for each message \( m \), the send-action of \( m \) precedes the receive-action of \( m \).

Formally, we define basic charts as follows.

Let \( M \) be a set of messages, and let the set of events be given as \( \Sigma \triangleq (M \times \{!, ?\}) \).

That is, an event \( e \) is either \( m! \) (indicating that message \( m \) is sent) or \( m? \) (indicating that \( m \) is received).

A lifeline \( l \) is a finite (possibly empty) sequence of events \( l \triangleq (e_1, e_2, ..., e_n) \). A basic chart \( C \) is an \( n \)-tuple of lifelines \( \langle l_1, ..., l_n \rangle \) with \( l_i = (e_{i1}, ..., e_{im}) \). We say that an event \( e \) occurs at the location \( (i, j) \) in chart \( C \) if \( e = e_{ij} \). An event occurrence \( o \triangleq (e, i, j) \) is a tuple consisting of an event \( e \) and the location of \( e \). We define \( \text{loc}(o) \triangleq (i, j) \) as the location of an event occurrence \( o \), and \( \text{lab}(o) \triangleq e \) be the event of \( o \). We denote the set of event occurrences appearing in \( C \) with \( \text{EO}(C) \).

A communication \( \langle (m!, i, j), (m?, i', j') \rangle \) in \( C \) is a pair of two event occurrences in \( C \) representing sending and receiving of the same message \( m \). We define \( \text{mat}(m!, i, j) \triangleq (m?, i, j) \) to match a sending event occurrence to a receiving event occurrence of the same communication. A communication does not have to be completely specified by a basic chart. That is, it is possible that only the sending event or the receiving event of a message appears in a basic chart. In addition, an event is allowed to occur multiple times in a basic chart, i.e., a basic chart can express that a message is repeatedly exchanged. However, each event occurrence is unique in a basic chart.

The partial relation induced by a chart \( C \) on \( \text{EO}(C) \) is formalized as follows.

1. for any \( 1 \leq xj < |l_{xi}| \) with \( l_{xi} \) being a lifeline in \( C \), it holds that \( (e, xi, xj) \prec (e', xi, (xj + 1)) \);
2. for any \( o \in S \), it holds that \( o \prec \text{mat}(o) \); and
3. \( \prec \) is the smallest relation satisfying 1 and 2.

We admit the non-degeneracy assumption proposed by Alur et. al. [2]: a basic chart cannot reverse the receiving order of two identical messages sent by some lifeline. Formally, a basic chart is degeneracy if and only if there exist two sending event occurrences \( o_1, o_2 \in S \) with \( o_1 \prec o_2 \) such that \( \text{lab}(o_1) = \text{lab}(o_2) \) and \( \text{mat}(o_1) \neq \text{mat}(o_2) \).

For a basic chart, event occurrences are allowed to be absent, i.e., it is possible that only a sending event or a receiving event of a message appears in a basic chart. Each event occurrence is unique in a basic chart.
With basic charts, a universal eLSC can be defined as follows. A universal chart in the eLSCs consists of two basic charts: a main-chart ($Mch$, drawn within a solid rectangle) and a pre-chart ($Pch$). There are two possibilities of pre-charts: “necessary pre-charts” (drawn within a solid hexagon) and “sufficient pre-charts” (drawn within a hashed hexagon). These two pre-charts are interpreted as a necessary condition and a sufficient condition for a main-chart, respectively. Intuitively, a universal chart with a necessary pre-chart specifies all traces such that, if contains a segment which is admitted by the pre-chart, then it must also contain a continuation segment (directly following the first segment) which is admitted by the main chart. On the other hand, a universal chart with a sufficient pre-chart specifies all traces such that, if contains a segment which is admitted by the main-chart, then the segment must (directly) follows a prefix segment which is admitted by the pre-chart. Formally, the syntax of eLSCs is as follows.

**Definition 3 (Syntax of universal eLSCs).** A universal eLSC is a tuple

$$Uch \triangleq (Pch, Mch, Cate)$$

with $Cate \in \{Suff, Nec\}$ denoting the category of the pre-chart. More specifically, the chart $(Pch, Mch, Suff)$ is with a sufficient pre-chart, and $(Pch, Mch, Nec)$ is with a necessary pre-chart.

We define PeLSCs by introducing condition structure and assignment structure into eLSCs.

An assignment structure is comprised of a function $v := s$ with $v$ being a variable and $s$ being the name of a parameter. The variable $v$ is evaluated to the value of a parameter name $p$. The function is surrounded by a rectangle with a sandglass icon at the top right corner. A condition structure is comprised of a proposition $prop$ surrounded by a rectangle. The proposition expresses the comparisons of parameter values. The notations for the assignment structure and the condition structure are shown in Fig. 2.

**Fig. 2.** Examples: an assignment structure (left) and a condition structure (right)

In a PeLSC, assignment structures and condition structures combine naturally with event occurrences. Intuitively, an assignment structure stores the value of a parameter carried by the combined event occurrence; and a condition structure expresses the features of a parameter carried by the combined event occurrence. Formally, the syntax of the two structures are given as follows.

**Definition 4 (Syntax of assignment and condition structures).** Let $Uch$ be an eLSC, $o \in EO(Uch)$ an event occurrence of $Uch$, $v$ a free variable, $s$ a nominal, and $prop$ a proposition. An assignment structure is defined as a tuple $assi \triangleq (v, s, o)$, where $s$ represents the name of of a parameter. A condition structure is defined as a pair $cond \triangleq (prop, o)$. 
With these structures, a PeLSC can be defined as follows.

**Definition 5 (Syntax of PeLSCs).** A PeLSC is defined as a tuple \( PU \triangleq (Uch, \text{COND}, \text{ASSI}) \), where \( Uch \) is an eLSC, and \( \text{COND} \) and \( \text{ASSI} \) are sets of condition structures and assignment structures, respectively.

There are two possible forms of propositions in condition structures, one is with free variables (denoted by \( \text{prop}(s_1, ..., s_n, v_1, ..., v_m) \)) and the other is without free variables (denoted by \( \text{prop}(s_1, ..., s_n) \)). These two forms are used to express relative parameter values and absolute parameter values, respectively. We divide the set \( \text{COND} \) of a PeLSC into two subsets \( \text{COND}_{FV} \) and \( \text{COND}_{NFV} \). The subset \( \text{COND}_{FV} \) is comprised of the set of condition structures with propositions of the form \( \text{prop}(s_1, ..., s_n, v_1, ..., v_m) \); and the subset \( \text{COND}_{NFV} \) is comprised of the set of condition structures with propositions of the form \( \text{prop}(s_1, ..., s_n) \). It holds that \( (\text{COND}_{FV} \cap \text{COND}_{NFV}) = \emptyset \) and \( (\text{COND}_{FV} \cup \text{COND}_{NFV}) = \text{COND} \).

Given a parametrized event trace, the proposition in a condition structure \( \langle \text{prop}, o \rangle \) is evaluated to a boolean value, according to the parameter values carried by events:

- the nominals \( s_1, ..., s_n \) in \( \text{prop} \) are replaced by the values of the parameters named by \( s_1, ..., s_n \) carried by the event \( \text{lab}(o) \); and
- the variables \( v_1, ..., v_m \) in \( \text{prop} \) are evaluated to the values from some event occurrences through the assignment structures \( \text{assi}(v_1, s_{x1}, o_{x1}), ..., \text{assi}(v_m, s_{xm}, o_{xm}) \).

In this paper, we require our PeLSCCs to satisfy an additional non-ambiguity assumption. We say a PeLSC is non-ambiguity, if for any condition structure with a proposition of the form \( \text{prop}(s_1, ..., s_n, v_1, ..., v_m) \), all free variable \( v_1, ..., v_m \) are evaluated to a certain value from a unique event occurrence. More formally, a PeLSC \( PU \triangleq (Uch, \text{COND}, \text{ASSI}) \) is non-ambiguity if and only if for any condition structure \( \text{condition} = \langle \text{prop}(s_1, ..., s_n, v_1, ..., v_m), o \rangle \) in the set \( \text{COND}_{FV} \) it holds that for any \( v_{xi} \in \{v_1, ..., v_m\} \) there exists an assignment structure \( \langle v_{xi}, s_{xi}, o' \rangle \rangle \in \text{ASSI} \) with \( o' \prec o \). With the non-ambiguity assumption, each proposition is able to be evaluated to a certain boolean value (true or false) over a deterministic parametrized event trace. To understand this assumption, consider the PeLSCs in Fig. 3. For the chart \( PU_2 \), the variable \( v \) has no value since there is no assignment structure to store it. For the chart \( PU_3 \), the variable \( v \) has two values stored by two assignment structures. Therefore, the condition structure cannot be evaluated to a certain boolean value for both of the two charts. For the chart \( PU_4 \), the values of variables \( v_1 \) and \( v_2 \) are from different event occurrences.

The non-ambiguity assumption is a strong assumption. For instance, the variables \( v_1 \) and \( v_2 \) in the chart \( PU_4 \) have certain values. However, the order of the two events \( !m_1 \) and \( !m_2 \), which are combined with the two assignment structures of \( v_1 \) and \( v_2 \), are uncertain. Since our monitors are generated by translating PeLSCs into HL, the size of the monitor is increased by expressing all possible executions of the pre-chart in the resulting formula. This will reduce the monitoring efficiency.
3.2 The Trace-based Semantics of PeLSCs

A PeLSC \((Uch, \text{COND}, \text{ASSI})\) defines a parametrized language (a set of parametrized event traces) that is an extension of the propositional language (i.e., a set of event traces) defined by \(Uch\). Intuitively, the parametrized language is comprised of all parametrized event traces such that the orders of events meets the partial order induced by \(Uch\), and all propositions in \(\text{COND}\) are evaluated to \(true\) with the values from the parameters carried by the events. A parametrized event trace \(\tau\) is admitted by a PeLSC \(PU\) if and only if \(\tau\) is in the parametrized language defined by \(PU\).

A set of sequences of event occurrences is defined by a basic chart \(C\) as follows:

\[
EOcc(C) \triangleq \{(o[1], o[2], \ldots, o[n]) \mid \{o[1], o[2], \ldots, o[n]\} = EO(C); n = |EO(C)|; \text{and for all } o[i], o[j] \in EO(C), \text{if } o[i] \prec o[j], \text{then } xi < xj\}.
\]

For languages \(L\) and \(L'\), let \((L \circ L')\) be the concatenation of \(L\) and \(L'\) (i.e., \((L \circ L') \triangleq \{(o) | o \in L \text{ and } o' \in L'\})\); and \(\overline{L}\) be the complement of \(L\) (i.e., for any \(\sigma \in \Sigma^*\), it holds that \(\sigma \in \overline{L}\) iff \(\sigma \notin L\)). The language of event occurrences for eLSCs is given as follows.

**Definition 6 (Semantics of universal eLSCs).** The language of event occurrences defined by an eLSC \(Uch = (Pch, Mch, Cate)\) is as follows

- \(EOcc(Uch) \triangleq EOcc(Pch) \circ EOcc(Mch)\), if \(Cate = \text{Suff}\);
- \(EOcc(Uch) \triangleq EOcc(Pch) \circ EOcc(Mch)\), if \(Cate = \text{Nec}\).

The evaluations of the propositions are formally defined as follows. Given a set \(\mathcal{P} = \{\{(s_1, d_1), \ldots, (s_n, d_m)\}\}\) of parameters and a proposition \(\text{prop}(s_y, \ldots, s_y)\), we define the evaluation of \(\text{prop}(s_y, \ldots, s_y)\) over \(\mathcal{P}\) (denoted with \([\mathcal{P} \vdash \text{prop}(s_y, \ldots, s_y)]\)) as follows.

\[
[\mathcal{P} \vdash \text{prop}(s_y, \ldots, s_y)] = \begin{cases} 
true & \text{if } \text{prop}(d_y, \ldots, d_y) \text{ is satisfied with } \{s_y, d_y\}, \ldots, \{s_m, d_m\} \subseteq \mathcal{P} \\
false & \text{otherwise}
\end{cases}
\]

For the propositions of the form \(\text{prop}(s_y, \ldots, s_y, v_{y1}, \ldots, v_{ym})\) in \(PU\), the values of the variables \(v_{y1}, \ldots, v_{ym}\) are from the assignment structures.
ASSI(vγ₁,...,vγm) ≜ \{\text{assi}(vγ₁),...,\text{assi}(vγm)\}, where \text{assi}(vγi) ≜ (vγi,sγi,o).

By \text{ac}(s_ξ₁,...,s_ξn,vγ₁,...,vγm) we denote the pair \((\text{prop}(s_ξ₁,...,s_ξn,vγ₁,...,vγm),\text{ASSI}(vγ₁,...,vγm))\). Given a pair \((\mathcal{P},\mathcal{P}')\) of two sets of parameters, the evaluation of the pair \text{ac}(s_ξ₁,...,s_ξn,vγ₁,...,vγm) over \((\mathcal{P},\mathcal{P}')\) is defined as follows.

- \([((\mathcal{P},\mathcal{P}′) \vdash \text{ac}(s_ξ₁,...,s_ξn,vγ₁,...,vγm))] = true, if \text{prop}(d_ξ₁,...,d_ξn,dγ₁,...,dγm) is satisfied with \{(s_ξ₁,d_ξ₁),...,(s_ξn,d_ξn)\} ⊆ \mathcal{P}′ and \{(s_γ₁,d_γ₁),...,(s_γn,d_γn)\} ⊆ \mathcal{P};
- \([((\mathcal{P},\mathcal{P}′) \vdash \text{ac}(s_ξ₁,...,s_ξn,vγ₁,...,vγm))] = false, otherwise.

A PeLSC \(PU\) defines all parametrized event traces \((\sigma,\rho)\) such that

- \(\sigma = (lab(o[1]),...,lab(o[n]))\) with \((o[1],...,o[n]) \in EOcc(Uch)\);
- for all \((\text{prop},o) \in \text{COND}_{N\text{FV}}\), if there exists \(z_i \in [1,n]\) with \(o[z_i] = o\), then \([[\rho[z_i] \vdash \text{prop}]] = true; and
- for all \((\text{prop}(s_ξ₁,...,s_ξn,vγ₁,...,vγm),o) \in \text{COND}_{\text{FV}}\) and \((vγ₁,sγ₁,o') \in \text{ASSI}(vγ₁,...,vγm)\), if there exists \(z_j,z_k \in [1,n]\) such that \(o[z_j] = o'\) and \(o[z_k] = o\), then \([[(\rho[z_j],\rho[z_k]) \vdash \text{prop}]] = true\).

By \(\text{PTra}(PU)\) we denote the set of parametrized event traces defined by \(PB\). Let \(\mathcal{E}(Uch)\) be the set of events appearing in an eLSC. We call each \(\epsilon_{Uch} \in (\Sigma \setminus \mathcal{E}(Uch))\) a stutter event, and \(\mathcal{P}_ε \in 2^{N\mathbb{D}}\) an arbitrary set of stutter parameters.

**Definition 7 (Trace based semantics for PeLSCs).** The parametrized language defined by a PeLSC \(PU\) is

\[\mathcal{PL}(PU) \triangleq \{(ε*,ε₁,ε₁,...,ε_n),((\mathcal{P}_ε,\mathcal{P}_1,...,\mathcal{P}_n,\mathcal{P}_ε))\},\]

where \([(ε*,ε₁,ε₁,...,ε_n)] \in \text{PTra}(PU), and \[(ε*,ε₁,ε₁,...,ε_n,ε*)] = [(\mathcal{P}_ε,\mathcal{P}_1,...,\mathcal{P}_n,\mathcal{P}_ε)], and ε* and \(\mathcal{P}_ε\) are finite (possible empty) sequences of stutter events and stutter parameters, respectively.

# 4 A Translation of PeLSCs into HL Formulae

In this subsection, we present a translation of PeLSCs into a subclass of the hybrid logic (HL) formulae. Whether an observation is admitted by a PeLSC can then be checked with the resulting formula.

The syntax and semantics of HL are given as follows.

**Definition 8 (Syntax of HL).** Given the finite set \(AP\) of atomic propositions, a set \(V\) of variables, the set \(\mathbb{Z}\) of integers, and a set \(N\) of nominals, the terms \(\pi\) and formulae \(\varphi\) of HL are inductively formed according to the following grammar, where \(x \in V, p \in AP, s \in N, r \in \mathbb{Z}\) and \(\sim \in \{<,=\}:

\[
\pi := x + r \mid r
\]

\[
\varphi := \bot \mid p \mid (\varphi₁ \Rightarrow \varphi₂) \mid (\varphi₁ \mathcal{U} \varphi₂) \mid (\pi₁ \sim \pi₂) \mid x \downarrow s, \varphi.
\]
Intuitively, an **HL** formula \( x \downarrow s.\varphi(x) \) is satisfied by a parametrized event trace \( \tau \triangleq (\sigma, \rho) \) if and only if \( \varphi(d) \) is satisfied by \( \sigma \) with \( (s, d) \in \rho[1] \). For instance, let \( t \) and \( id \) be parameters representing time stamps and clients' ID, respectively, a formula

\[
\Box x \downarrow t.y \downarrow id.(\text{login} \Rightarrow \Diamond x' \downarrow t.y' \downarrow id.(\text{logout} \land (y' = y) \land (x' < 200 + x)))
\]

expresses the property \( \text{Pro} \).

Assume that \( \mathcal{E} \) is a function \( \mathcal{E} : V \rightarrow \mathbb{Z} \) for assigning free variables in the domain of integers \( \mathbb{N}_{\geq 0} \) such that \( \mathcal{E}(x + d) = \mathcal{E}(x) + d \) and \( \mathcal{E}(d) = d \). Given a variable \( x \) and a natural number \( d \), we denote \( \mathcal{E}[x := d] \) for the evaluation \( \mathcal{E}' \) such that \( \mathcal{E}'(x) = d \), and \( \mathcal{E}'(y) = \mathcal{E}(y) \) for all \( y \in V \setminus \{x\} \). The **HL** is defined on parametrized event traces as follows.

**Definition 9 (Trace-based Semantics for HL).**

Let \( \tau \triangleq (\sigma, \rho) \) be a parametrized event trace with \( \sigma \triangleq (e[1], e[2], \ldots) \) being an event trace and \( \rho \triangleq (P[1], P[2], \ldots) \) being a parameter trace. Let \( i \in \mathbb{Z}_{\geq 0} \) be a position, \( p \) a proposition, \( s \) a nominal, \( d \) a value in the domain of parameters, and \( \varphi_1 \) and \( \varphi_2 \) any **HL** formulae. The satisfaction relation \( (\tau, i, \mathcal{E}) \models \varphi \) is defined inductively as follows:

\[
(\tau, i, \mathcal{E}) \models p \text{ iff } p \in e[i]; \\
(\tau, i, \mathcal{E}) \models (\varphi_1 \Rightarrow \varphi_2) \text{ iff } (\tau, i, \mathcal{E}) \models \varphi_1 \text{ implies } (\tau, i, \mathcal{E}) \models \varphi_2; \\
(\tau, i, \mathcal{E}) \models (\varphi_1 \cup \varphi_2) \text{ iff there exists } j > i \text{ with } (\tau, j, \mathcal{E}) \models \varphi_2 \text{ and for all } i < j' < j \text{ it holds that } (\tau, j', \mathcal{E}) \models \varphi_1; \\
(\tau, i, \mathcal{E}) \models \pi_1 \sim \pi_2 \text{ iff } \mathcal{E}(\pi_1) \sim \mathcal{E}(\pi_2); \\
(\tau, i, \mathcal{E}) \models x \downarrow s.\varphi \text{ iff } (\tau, i, \mathcal{E}[x := d]) \models \varphi, \text{ where } (s, d) \in P[i].
\]

As usual, \( \tau \models \varphi \text{ iff } (\tau, 1, \mathcal{E}) \models \varphi. \)

We now show how to translate a PeLSC into an **HL** formula to check whether a parametrized trace is admitted. The translation is developed on basis of the translation form an eLSC \( Uch \) into an LTL formulae \( \varphi(Uch) \) as shown in [14]. Here we concern the translation of the introduced assignment and condition structures for PeLSCs.

A PeLSC is comprised of a universal eLSC \( Uch \), a set \( \text{COND} \) of condition structures and a set \( \text{ASSI} \) of assignment structures. Propositions appearing in a PeLSC are specified by the comparisons of terms, i.e., \( \pi_1 \sim \pi_2 \). According to the subset \( \text{COND}_{NFV} \) and \( \text{COND}_{PV} \), the following formulae are defined.

- Let \( e \) be an even in \( Uch \) combined with a constraint structure \( \text{cond} = (\text{prop}, o \text{ from the set } \text{COND}_{NFV} \text{ with prop}(s_1, \ldots, s_n) \text{ and lab}(o) = e). \) The condition structure is translated into an **HL** formula

\[
o(\text{cond}) \triangleq \Box (x_1 \downarrow s_1. \cdots x_n \downarrow s_n (e \Rightarrow \text{prop}(x_1, \ldots, x_n))).
\]

The formula specifies that whenever the event \( e \) occurs, then the proposition must be evaluated to \( \text{true} \) with values of the parameters named by \( s_1, \ldots, s_n \) carried by \( e \). I.e., if the event occurs at a position \( z_1 \) of the trace, the proposition \( \text{prop}(d_1, \ldots, d_n) \) is true with \( \{(s_1, d_1), \ldots, (s_n, d_n)\} \subseteq \rho[z_1]. \)
Theorem 1. The complexity of the word-problem of \( \text{HL} \) is linear with respect to the size of input traces.

Proof. Given an \( \text{HL} \) formula \( \varphi = x \downarrow s.\psi(x) \) and a parametrized event trace \( \tau = (\sigma, \rho) \). The trace \( \tau \) satisfies \( \varphi \) if and only if \( \sigma \models \psi(d) \) with \( (s, d) \in \rho[1] \). Since \( d \) is a certain integer value, the sub-formulae of the comparisons of terms in \( \varphi \) is able to be directly evaluated a boolean value true or false. Therefore, the process of checking an \( \text{HL} \) formula over a parametrized event trace \( \tau \) is essentially the same with the process of checking an LTL formula over an event trace \( \sigma \). Since the complexity of checking whether or not a trace \( \sigma \) satisfies an LTL formula is linear with respect to the size of the trace \( \sigma \), the complexity of the word-problem of \( \text{HL} \) is linear with respect to the length of the input parametrized event traces.

Corollary 1. The complexity of PeLSCs is linear with respect to the size of traces.

According to the corollary, the language of PeLSCs is feasible for runtime verification implementations, especially for on-line monitoring.

We implement the algorithms in Maude [17], which provides a formula rewriting environment for monitoring. The implementation is evaluated on several benchmarks.
The monitoring efficiency for the property \textbf{P3} is shown in Fig. 4. The property \textbf{P3} is comprised of an eLSC and two condition structures with assignments (i.e., with free variables). Fig. 4(a) shows the monitoring efficiency for the condition structures, and Fig. 4(b) shows the monitoring efficiency for the eLSC. In this monitoring implementation, the most rewrites are spent on monitoring the condition structures with free variables.

![Monitoring Efficiency for P3](image)

(a) Parametric requirements  
(b) Temporal requirements

**Fig. 4. Monitoring efficiency for P3**

5 Conclusion

In this paper, we defined PeLSCs for parametric properties by introducing assignment and condition structures into LSCs. With these structures, PeLSC can be interpreted over parametrized event traces. The language can than intuitively express requirement of data (e.g., values of time or other variables) carried by events. We developed translation from PeLSCs into \textbf{HL} for monitoring. We prove that the complexity of the word problem of PeLSCs is linear if propositions in the condition structures only express comparisons of terms.

There are several interesting topics for future work. Firstly, in this paper, we only concerned comparisons of terms in the condition structures. It is interesting to find out whether or not the PeLSC is still feasible for monitoring if the expressiveness of conditions is extended by, e.g., introducing quantifiers \(\forall\) and \(\exists\). Secondly, since the sizes of resulting formulae are often large, translating PeLSC into \textbf{HL} formulae is not an efficient way for monitoring. Therefore, we are currently developing a more efficient implementation, which can check PeLSC specifications directly. Last but not least, the synthesis problem of PeLSC based monitors is left open in this paper. As PeLSCs have features of the first order logic, the existing LSC synthesising techniques cannot handle this problem.
References

Remarks on Memory Consistency Description

Ludwik Czaja
Vistula University, Warsaw
and
Institute of Informatics, The University of Warsaw
lczaja@mimuw.edu.pl

Abstract. Two observations in the matter of pictorial as well as formal presentation of some consistency in distributed shared memory are made. The first concerns geometric transformation of line segments and points picturing read/write operations, the second - converting partial order of the operations into linear order of their initiations and terminations. This allows to reduce serialization of the read/write operations as a whole to permutations of their beginnings and ends. Some draft proposals are introduced.

1 Introduction

Description of a consistency model for a distributed shared memory (DSM) requires determining what partial orders between constituents of sequential processes are admissible. The constituents need duration, some of them possibly long - like the update of all replicas of variables. Customarily, a process is pictured as a sequence of line segments or points corresponding to read/write operations, and parallel run as a number of such sequences put on rightwards directed parallel straight lines representing passage of global time. But a certain vagueness and difficulty yields such presentation. What is the right end of a segment corresponding to write operation? Is it a time instant when update of all replicas of a variable being written is completed, or memory manager’s response that write request was accepted? Not always the partial order of operations can be converted into a linear order retaining the result of execution, required by some kinds of consistency. For these reasons finer granularity of constituents of processes is proposed: instead of read/write operations as a whole, timeless events of their initiations (invocations) and terminations. In such semantic model, each parallel execution may be represented as a sequence of events, in which independent events may occur in arbitrary order. Furthermore, decision on whether or not a parallel execution can be converted into equivalent sequential, reduces to finding a suitable permutation of the sequence. Another observation is that arrangement of partially ordered executions of read/write operations into a linear order, in geometrical terms means a transformation of line segments representing the operations: shifting them along the time axis.

A schematic image of a system of two computers with DSM is in Fig. 1.1 This virtual memory - a common address space - is here seen as a union of local memories of computers.
2 Parallel Executions as Succession of Global States

Consider the following system of two computers running programs in parallel with initial values of variables $a = b = 0$. Reading their values is associated with sending them to print.

$$
\begin{align*}
\text{computer 1} & : & a & := 1; & b & := 1; & \text{print}(b); \\
\text{computer 2} & : & \text{print}(a);
\end{align*}
$$

Neither the entire program statements, nor the access operations to the shared variables $a, b$ are assumed to be atomic (indivisible). One of possible executions of the system, presented as a succession of its global states 1 - 4, is shown in Table 1. The states are chosen to display a progress of the system activity. Thus, in a sense, the choice is arbitrary: adapted to a generality level suitable for presentation of a considered problem. Here, writing to shared variables $a, b$ (state 1) and reading their values (along with sending to print - state 2), is performed in parallel by both computers. The dashed arrows stretching across successive states represent the progress of data transmission from one computer to the other.

Two remarks on this execution:

- A variable has a location in the local memory of each computer using this variable: a paging (or cache memory) mechanism ensuring a local accessibility of the variable is assumed (its functioning is not displayed in Table 1). That is why replicas of one variable have locations in local memories of computers using this variable.

- Every computer reads value of a variable before it is updated in all replicas. That is why printouts are $a = 0, b = 0$, but not $a = 1, b = 1$ - what would be the case if progress of data transmission outdistanced reading of values of $a$ and $b$ (transmissions racing)
Table 1. State 1: local update of values of variables $a, b$ and start of their transmission. State 2: local readout of values of variables $a, b$ with sending them to print and their transmission in progress. State 3: end of transmission of the value of variable $a$ and transmission of the value of variable $b$ in progress. State 4: end of transmission of the value of variable $b$. 
3 Geometric Presentation of Parallel Executions

A set of read/write operation occurrences in a single computer process is called a history of memory access of this process (history of the process in short). Customarily, it is pictured as a set of line segments on a rightwards directed straight line - a global time axis. Each segment represents a time interval between initiation and termination of a read or write execution. Initiation, termed an invocation of the memory access, is shown as the left-most point of the segment. It represents an instant of global time in which request for the operation occurs in the process. Similarly, the right-most point of the segment shows the termination of a read or write and for a read it represents an instant of global time when a value being read is fully fetched from memory to processor. However for a write, the right-most point of the segment may be understood as an instant of global time in which either:

1) all replicas of a shared variable’s value are updated by the value being written, or
2) a response (acknowledgment of invocation) from the subsystem managing memory access is received by the process.

A placement of the line segments on the global time axis depends on the case taken into account. In the case (1), the history of process is not always a sequence but sometimes merely a set of segments, i.e. some segments may overlap - they are partially ordered in time. This happens because transmission of data through the network not always makes the process suspend: the subsystem managing memory access may proceed in parallel with another activity of the process. In the case (2), the history of process is always a sequence of segments - they are totally ordered. Therefore, in the case (1), the execution shown in Table 1 is graphically represented in Fig.3.1, while in the case (2) - in Fig.3.2.
4 Sequential and Strict Consistency/Inconsistency - Informal

The system \{computer 1, computer 2\} in Fig. 2.1 may be executed in many ways: a succession of its R/W operation instances in global time may be arbitrary unless restrictions on the activity of subsystems managing of memory access is imposed. Various executions may render various outcomes, i.e. printouts and memory content. Consider the following three examples:

- The DSM managed by subsystems permitting of execution like in Table 1 (geometrically Figures 3.1, 3.2) is a sequentially inconsistent memory. The four read/write operations cannot be arranged in one sequence preserving their order of execution in individual programs and result of computation: these two requirements are in contradiction! Indeed, in such arrangement, \(R(b, 0)_1\) should precede \(W(b, 1)_2\) and \(R(a, 0)_2\) should precede \(W(a, 1)_1\) which violates the order of their execution specified by programs. The segments representing the operations cannot be arranged on the global time axis so that the two requirements are fulfilled.

- Let us modify the above execution replacing \(R(b, 0)_1\) with \(R(b, 1)_1\). Now, the operations can be arranged in one sequence fulfilling the two above requirements: \(W(b, 1)_2\) \(R(a, 0)_2\) \(W(a, 1)_1\) \(R(b, 1)_1\). The corresponding segments have been suitably shifted (a geometric transformation). The DSM managed by subsystems which permit such executions but prohibit such as in the latter example, is a sequentially consistent memory.

- Finally, let us modify the latter execution replacing \(R(a, 0)_2\) with \(R(a, 1)_2\). The sequence of operations fulfilling the two requirements is now, e.g. \(W(a, 1)_1\) \(W(b, 1)_2\) \(R(b, 1)_1\) \(R(a, 1)_2\). Moreover, the sequence preserves succession of operation invocations in global time. The DSM managed by subsystems which permit such executions but prohibit such as in the first example, is a strictly consistent memory.

In the first example, no requirement on conformity (order similarity) between a succession of read/write execution specified by individual programs and order of these operations during the system concurrent run, is imposed. In the second, such conformity is required as well as identical outcome of sequential and concurrent executions. In the third example, apart from requirements such as in the latter, the conformity between the global time order of invocations of read/write operations and order of their instances in sequential arrangement should be possible.

Bearing in mind the examples, one may express not too formally both properties of DSM:

**Sequential consistency.** For any parallel execution there exists a sequential execution i.e. a sequence of read/write operations where their succession is the same as specified by individual programs; furthermore, each readout of a variable fetches a value assigned to this variable by a write operation, which precedes the readout in this sequence, and assignment of different value to this variable does not occur inbetween.

**Strict consistency.** In any parallel execution, each readout of a variable fetches a value assigned to this variable by a write operation performed before this readout in global time, and assignment of different value to this variable has not been performed inbetween.
Therefore, the meaning of words "precede", "inbetween" etc. is relative to a context of their usage: they may refer to the global time or to a position in a sequence. This must be specified whenever such words are used.

Remarks

(1) Sequential and strict consistency or inconsistency happens also if arrangement of access operations to one variable only is considered. For example, the system in Fig.4.1 may be executed as in Fig.4.2. But the read/write operations cannot be arranged in one sequence satisfying property of sequential consistency. Here too, reading a value of variable \( a \) is associated with sending it to print.

\[
\begin{align*}
\text{computer 1:} & \quad \text{computer 2:} \quad \text{computer 3:} \quad \text{computer 4:} \\
 a := 1; & \quad a := 2; \quad \text{print}(a); \quad \text{print}(a); \\
\ldots & \quad \ldots \quad \ldots \quad \ldots \\
\ldots & \quad \ldots \quad \text{print}(a); \quad \text{print}(a);
\end{align*}
\]

Fig.4.1

However in another execution, which differs from the one in Fig.4.2 so that \( R(a, 2)_4 \) is replaced with \( R(a, 1)_4 \), the read/write operations can be arranged in one sequence: \( W(a, 2)_2, R(a, 2)_3, W(a, 1)_1, R(a, 1)_4, R(a, 1)_3, R(a, 1)_4 \). Such execution satisfies property of sequential consistency. Execution in which all read operations fetch value 2 would allow to arrange all the read/write operations in one sequence in which their succession is the same as their order of execution in global time. Such sequence satisfies property of strict consistency.

(2) Geometrically, arrangement of read/write operations in a sequence satisfying property of sequential consistency causes a transformation of line segments: a shift along the global time axis. For instance in the modified Fig.4.2, the segment \( W(a, 1)_1 \) would be shifted rightwards, so that this write is performed later than \( R(a, 2)_3 \) and earlier than \( R(a, 1)_4 \); a delay of execution of \( W(a, 1)_1 \) took place.

(3) Arrangement of read/write operations in one sequence boils down to extension of their partial order to linear. The examples demonstrate that this is not possible for some executions when restrictions defining sequential or strict consistency are imposed. Without any restrictions it is always possible, also for infinite history of processes. This
is widely known due to a very general theorem: for any partial order relation \( \preceq \) on a set \( X \) there exists a linear order \( < \) on \( X \) such that \( \preceq \subseteq < \) [KM 1976], [Mar 1996].

(4) The unpredictable duration of operations on memory ("coarse granularity") and their presentation as line segments, makes images of parallel execution somewhat counterintuitive, especially when right-most points of the segments represent acknowledgment of invocations (case 2 in Section 3) - not update of all replicas. Worse, not every parallel execution can be described in terms of interleaving of sequences of read/write operations: not always the interleavings exist for some consistency models. The segments are partially ordered, so, one cannot speak of their permutations, what facilitates formal description of some consistency models proposed in Section 6. That is why the (timeless) events of initiation and termination of read/write operations ("fine granularity"), not the operations at the whole length, will be taken as atomic (indivisible) building units of processes in the next Section. With such fine granularity, all events can be ordered into global sequences - interleavings of local histories of memory operations. Parallel executions will be represented by such interleavings, where independent events may occur in the arbitrary order, thus one execution may be represented by several interleavings (note that such approach, in the abstract setting, was a basis for Mazurkiewicz traces [Maz 1987], [TBT 1995] THE BOOK OF TRACES, edited by V. Diekert and G. Rozenberg, World Scientific Publ. Co. 1995).

5 Events of Initiation and Termination of Read/Write

Let us admit the following denotations and assumptions:

(1) Initiation of reading value \( \alpha \) of variable \( x \) by computer number \( j = 1, 2, ..., N \) is denoted by \( R(x, \alpha)_j \) and termination by \( R(x, \alpha)_j \); similarly of writing: \( \overline{W}(x, \alpha)_j \), \( W(x, \alpha)_j \). Events \( R(x, \alpha)_j \) and \( \overline{W}(x, \alpha)_j \) are interpreted as invocations for performing the operations. Event \( R(x, \alpha)_j \) is interpreted as reception of value \( \alpha \) of variable \( x \) and \( \overline{W}(x, \alpha)_j \) as completion of updating of all replicas of \( x \) with the value \( \alpha \). A computer initiates and terminates the reading of a variable with the same value, and this concerns also the writing.

(2) Subsystems managing of memory access ensure serialization of initiation and termination events, thus their linear order. In any such sequence, initiation of an operation must precede its termination. Such model clearly exhibits arrangement of access operations on the global time axis. For example, the following sequence of events:

\[
\overline{W}(a, 1)_1 \ W(a, 2)_1 \ R(a, 2)_3 \ R(a, 2)_4 \ W(a, 1)_1 \ R(a, 1)_4 \ R(a, 1)_3 \ R(a, 1)_4 \ W(a, 2)_2 \ R(a, 2)_4 \ R(a, 1)_3 \ R(a, 2)_4
\]

shows succession in global time of initiations and complete terminations of read/write operations during parallel execution depicted in Fig.4.2. No sequence of read/write operations as a whole may yield the same result.

Figure 5.1 shows the same execution as Figures 3.1, 3.2, but with events of initiation and termination of read/write. Events \( W(a, 1)_1, W(b, 1)_2 \), denote completion of updating all replicas of \( a \) and \( b \) with the value 1.
The nested bracket structures represent both histories of memory access. Parallel execution progresses e.g. in the following interleaving of the histories:

$W(a,1) \quad R(b,0) \quad R(b,0) \quad W(a,1)$

$W(b,1) \quad R(a,0) \quad R(a,0) \quad W(b,1)$

with the nested bracket structure $(((()))$. For the memory coherence, the sequence of events should be rearranged so that its fragment from $R(a,0)\_2$ to $R(a,0)\_2$, thus the entire operation $R(a,0)\_2$, should precede entire operation $W(a,1)\_1$ and $R(b,0)\_1$ should precede $W(b,1)\_2$. On the other hand, for not violating the order of the individual programs run, $W(a,1)\_1$ should precede $R(b,0)\_1$ and $W(b,1)\_2$ should precede $R(a,0)\_2$ in this rearranged sequence. This is not possible, thus the execution in Fig.5.1 does not fulfill requirements of sequential consistency.

Modifying the execution in Fig.5.1 by replacing $R(b,0)\_1$, $R(b,0)\_1$ with $R(b,1)\_1$, $R(b,1)\_1$ accordingly, the sequential consistency is obtained by arrangement:

$W(b,1)\_2 \quad W(b,1)\_2 \quad R(a,0)\_2 \quad R(a,0)\_2 \quad W(a,1)\_1 \quad R(b,1)\_1 \quad W(a,1)\_1 \quad W(b,1)\_2 \quad R(b,1)\_1 \quad R(b,1)\_1$

i.e. permutation of the given sequence. The permuted sequence has the flat bracket structure $(()(())())$.

Taking timeless events (points), but not time consuming entire operations, (line segments) as elementary indivisible objects means usage of a linear instead of partial order for description of global system behaviours: the so-called "true concurrency" is simulated by nondeterminism. The example shows that a decision whether or not a given event-sequence can be transformed into a sequence fulfilling the property of sequential or strict consistency, reduces to finding of its suitable permutation. This suggests using this observation in a certain formalization of some consistency models presented in the next Section.

6 Some Consistency Models in the Interleaving Semantics

Let us admit the following denotations:

1. $S = \{P_1, P_2, \ldots, P_N\}$ – a system of sequential programs with DSM, performing in parallel by computers numbered $1, 2, \ldots, N$.

2. $V$ – set of variables used by the programs and allocated in DSM.

3. $D$ – set of values, the variables may assume.
Sequential and strict consistency: A global history \( Q = q_1q_2 \ldots q_n \in Ev^* \) uniquely determines the memory state (content) and printouts yielded by \( Q \). The memory state is a set of pairs \( \sigma(Q) = \{(x, \alpha) : x \in V, \alpha \in A\} \) such that: 

(i) If \( W(x, \alpha)_j \) or \( R(x, \alpha)_j \) occurs in \( \pi(Q) \) then it is immediately preceded by respective \( W(x, \alpha)_l \) or \( R(x, \alpha)_l \), 

(ii) Events \( W(x, \alpha)_j \) and \( R(x, \alpha)_j \) occur in the sequence \( \pi(Q) \) in the same order as in the local history of program \( P_j \); by virtue of (i) operations \( W(x, \alpha)_j \) and \( R(x, \alpha)_j \) are performed in the system \( S \) in the same global time order as in program \( P_j \). 

(iii) If an event \( R(x, \alpha)_j \) occurs in \( \pi(Q) \) then \( \alpha \) is an initial value or \( R(x, \alpha)_j \) is preceded in \( \pi(Q) \) by a certain event \( W(x, \beta)_k \) with no event \( W(x, \beta)_l \) in between, where \( \beta \neq \alpha \), \( k, i = 1, 2, ..., N \). 

The system \( S \) obeys the principle of sequential consistency of DSM iff \( S \) admits only global histories \( Q \) fulfilling the property of sequential consistency. In short: the memory managed by \( S \) in sequentially consistent.

The strict consistency is obtained if (i), (ii), (iii) are satisfied for the non-rearranging (identity) permutation \( \pi(Q) = Q \).

Remarks

1. The global history \( Q = q_1q_2 \ldots q_n \in Ev^* \) uniquely determines the memory state (content) and printouts yielded by \( Q \). The memory state is a set of pairs \( \sigma(Q) = \{(x, \alpha) : x \in V, \alpha \in A\} \) such a relation \( \sigma(Q) \subseteq V \times A \). If for each closed \( Q \) relation \( \sigma(Q) \) is a function \( \sigma(Q) : V \rightarrow A \) (i.e. \( x = y \) implies \( \sigma(Q)(x) = \sigma(Q)(y) \): values of all replicas of each variable are identical), then the memory is coherent. Sequentially consistent memory is coherent. Memory incoherence (lack of integrity) arises e.g. in effect of parallel execution shown in Fig.4.2. The execution yields the DSM content: in memory of computers 1,2,4, the value of \( a \) is 2 and of computer 3 is 1. The memory state is then \( \sigma(Q) = \{(a, 1), (a, 2)\} \), which is not a function.

2. It is known that in general a decision whether or not a given finite execution fulfills the sequential consistency property is intractable. It is evident in the model presented here: a search for a permutation satisfying (i),(ii),(iii) is required. A good many research tackled this task with the same answer, unless additional information is supplied about the system behaviour. Some examples of this research are [GK 1992], [CLS 2005], [WYTC 2011].
3. A challenge is to extend the consistency concept to infinite executions. For such executions some problems, e.g. starvation are explored, but (to my knowledge) not memory consistency models. Note that space of infinite executions is non-enumerable.

4. Strict (unacceptable for efficiency) and sequential consistency bring nearer the DSM system to the multiprocessor with one physical shared memory with direct access. Nonexistence of such memory is transparent for the DSM users. Applications where efficiency is more crucial than preservation of some execution order in individual programs, may tolerate more liberal models than sequential consistency, the natural model for users and more acceptable than strict consistency.

Causal consistency

In any execution, if an effect of an update (write) operation depends on another update (in the same or different process and of the same or different variable), then in every process, the global time order of readouts of the updated variables should be the same as of these updates.

Let us define a relation of causal dependency in the set $Ev : \rightsquigarrow \subseteq Ev \times Ev$. For events $p, q$ two auxiliary primary relations are admitted:

1. If $p$ precedes $q$ in the same process then $p \xrightarrow{\text{process}} q$

2. If event $q = W(x, \alpha)_i$ terminates reading value $\alpha$ of variable $x$ and $\alpha$ was assigned to $x$ by a write operation completed with event $p = W(x, \alpha)_j$ then $p \xrightarrow{\text{readout}} q$

Causal dependency $\rightsquigarrow$ is the least (wrt. $\subseteq$) relation such that:

(i) if $p \xrightarrow{\text{process}} q$ or $p \xrightarrow{\text{readout}} q$ then $p \rightsquigarrow q$

(ii) if $p \rightsquigarrow q$ and $q \rightsquigarrow r$ then $p \rightsquigarrow r$

If $p \rightsquigarrow q$ then $p$ is a cause of $q$ and $q$ is an effect of $p$. The events are independent if neither $p \rightsquigarrow q$ nor $q \rightsquigarrow p$, written $p \perp q$.

Let $R = \{R(x, \alpha)_j\}, W = \{W(x, \alpha)_j\}, j \in \{1, 2, \ldots, N\}, x \in V, \alpha \in D$.

Let $Q = q_1q_2\ldots q_n \in Ev^*$ be a closed history of execution of the system $S$. $Q$ is causally consistent iff for any $q_i, q_j \in W$ in $Q$ with $q_i \rightsquigarrow q_j$, and for any $q_k, q_l \in R$ in $Q$, the following holds: if $q_k \xrightarrow{\text{process}} q_l$ ($q_k, q_l$ are in the same process) and $q_i \xrightarrow{\text{readout}} q_j$, and $q_j \xrightarrow{\text{process}} q_i$; but if for some $q_i, q_j \in W$, $q_k, q_l \in R$ in $Q$, the relations $q_i \rightsquigarrow q_j$, $q_k \xrightarrow{\text{process}} q_l$, $q_i \xrightarrow{\text{readout}} q_j$, and $q_j \xrightarrow{\text{process}} q_i$ hold then $Q$ is causally inconsistent. In symbols:

$$\forall q_i, q_j \in W [q_i \rightsquigarrow q_j \Rightarrow \forall q_k, q_l \in R ((q_k \xrightarrow{\text{process}} q_l \land q_i \xrightarrow{\text{readout}} q_j) \Rightarrow q_k \xrightarrow{\text{process}} q_l)]$$

The system $S$ obeys the principle of causal consistency of DSM iff $S$ admits only global causally consistent histories $Q$. In short: the memory managed by $S$ is causally consistent.

Example. Let

$$Q = \ldots W(x, 9)_1 \ldots W(x, 9)_2 \ldots W(y, 3)_2 \ldots W(y, 3)_3 \ldots W(y, 3)_4 \ldots W(x, 9)_4$$

$q_1$ $q_2$ $q_3$ $q_4$ $q_5$ $q_6$ $q_7$
This causally inconsistent execution is presented graphically in Fig.6.1. By definition of causality relation \( q_1 \leadsto q_3 \) holds (suppose, for instance, that process \( P_2 \) after having read value 9 of variable \( x \), computes \( \sqrt{9} = 3 \) and writes 3 to \( y \); initiations of read/write operations are not shown for simplicity).

Conditions for causal consistency of a history \( Q \) may be schematically presented by diagram in Fig.6.2a. For demonstrating inconsistency in suffices finding events \( q_1, q_3, q_6, q_7 \) such that there exists a diagram in Fig.6.2b. The diagrams show relations between events in \( Q \).

**PRAM (Pipelined Random Access Memory) consistency**

In any execution, if two update (write) operations are in the same process, then in every process, the global time order of readouts of these updated variables should be the same as of the updates.

Let \( Q = q_1q_2\ldots q_n \in E^* \) be a closed history of execution of the system \( S \). \( Q \) is **PRAM-consistent** iff for any \( q_i, q_j \in W \) in \( Q \) with \( q_i \xrightarrow{\text{process}} q_j \), and for any \( q_k, q_l \in R \) in \( Q \), the following holds: if \( q_k \xrightarrow{\text{process}} q_l \) (\( q_k, q_l \) are in the same process) and \( q_i \xrightarrow{\text{readout}} q_k \) and \( q_j \xrightarrow{\text{readout}} q_l \) then \( q_k \xrightarrow{\text{process}} q_i \); but if for some \( q_i, q_j \in W \), \( q_k, q_l \in R \) in \( Q \),
the relations $q_i \xrightarrow{\text{process}} q_j$, $q_k \xrightarrow{\text{process}} q_l$, $q_i \xrightarrow{\text{readout}} q_k$ and $q_j \xrightarrow{\text{readout}} q_l$ hold then $Q$ is PRAM-inconsistent.

In symbols:

\[
\forall q_i, q_j \in W \left[ q_i \xrightarrow{\text{process}} q_j \Rightarrow \forall q_k, q_l \in R \left( q_k \xrightarrow{\text{process}} q_l \land q_i \xrightarrow{\text{readout}} q_k \land q_j \xrightarrow{\text{readout}} q_l \right) \right]
\]

The system $S$ obeys the principle of PRAM-consistency of DSM iff $S$ admits only global PRAM-consistent histories $Q$. In short: the memory managed by $S$ is PRAM-consistent.

Note that causally inconsistent execution shown in Fig.6.1 is PRAM-consistent. The formal definitions of the four memory consistency models clearly set up them into the known hierarchy: strict $\subset$ sequential $\subset$ causal $\subset$ PRAM.

References


[WYTC 2011] Weiwu Hu; Yunji Chen; Tianshi Chen; Cheng Qian; Lei Li, Linear Time Memory Consistency Verification, Computers, IEEE Transactions on, On page(s): 502 - 516 Volume: 61, Issue: 4, April 2011
The Method for Describing Changes in the Perception of Stenosis in Blood Vessels Caused by an Additional Drug

Sylwia Buregwa-Czuma¹, Jan G. Bazan¹, Lech Zareba², Stanisława Bazan-Socha³, Przemysław W. Pardel¹, Barbara Sokolowska¹, and Łukasz Dydo¹

¹ Interdisciplinary Centre for Computational Modelling, University of Rzeszow Pigonia 1, 35-310 Rzeszow, Poland
² Faculty of Mathematics and Natural Sciences, University of Rzeszow Pigonia 1, 35-310 Rzeszow, Poland
³ II Department of Internal Medicine, Jagiellonian University Medical College Skawinska 8, 31-066 Krakow, Poland

Abstract. The decision making depends on the perception of the world and the proper identification of objects. The perception can be modified by various factors, such as drugs or diet. The purpose of this research is to study how the disturbing factors can influence the perception. The idea was to introduce the description of the rules of these changes. We propose a method for evaluating the effect of additional therapy in patients with coronary heart disease based on the tree of the impact. The leaves of the tree provide cross-decision rules of perception changes which could be suggested as a solution to the problem of predicting changes in perception. The problems considered in this paper are associated with the design of classifiers which allow the perception of the object in the context of information related to the decision attribute.

Key words: classification, perception interference, cross-decision rules, tree of impact

1 Introduction

One of the aspects of the data mining is learning about the surrounding world by assigning meanings to received impressions, i.e. information provided by sensors. The way in which we perceive and interpret the real world significantly determines the identification of the objects and their classification, which affects the decision-making. The classifiers, based on the available features (conditional attributes), describe the value of the decision attribute and may be treated as approximate descriptions of concepts (decision classes)⁴,⁵. Therefore, the classifiers allow the perception of the tested object in the context of information related to the value of the decision attribute.

As a perception we understand a cognitive process that involves assigning the meanings to the received information [12]. We consider the perception associated with medical problems, namely the problem of the coronary artery stenosis in patients with stable coronary heart disease (CHD, see Section 2).

We study how the disturbing factors can influence the perception of stenoses in blood vessels. We show that some disturbing factors can be managed using data mining algorithms based on well-known statistics combined with cross-decision rules. Our
116

approach is illustrated using data representing medical treatment of the patients with stable CHD. The dataset, collected by the Second Department of Internal Medicine, Collegium Medicum, Jagiellonian University, Krakow, Poland, relates to 70 patients subjected to elective coronary angiography with possible percutaneous angioplasty. The decision problem, however, requires approximation of especially designed complex decision attribute, corresponding to the analysis of perception interference.

1.1 Perception and Classification

There are a number of factors that can modify the behavior of the tested object and change the way it is perceived, although generally the state of the object is not changed. The state is stable in the sense that its affiliation to the concept does not change. Sometimes we have no influence on the disturbing factor, for example, in the case of the environment or the weather. But sometimes it may be intentionally introduced or even managed, if we know how to do it (eg. drugs, type of therapy, diet).

If we assume that the perception is achieved by the classifier, this means that if a disturbing factor occurs, the classifier must be redesigned. In this situation, one can build a classifier using additional conditional attribute, representing the information about the occurrence of the disturbing factor. As a result, the classifier can be effective both when the agent is present or when it is not. It is also possible to construct two classifiers: the first one when there is no disturbing factor and the other one in the presence of it.

Another issue is, however, the question how does the disturbing factor change the perception of test objects using the classifier and whether the change affects all test subjects in the same way? If we could get to know the rules of these changes, this factor could be used to control the object perception and indirectly its behavior.

It is known from practical observations, that disturbing factors may often cause various changes in the perception of the observed object. For example, consider the situation when perception refers to the condition of the patient described by four states: A, B, C, D, associated with the severity of disease which may be life-threatening. The condition A indicates the lowest severity of the disease, B - slightly more advanced disease, C - yet slightly more and D - the most advanced disease. Suppose also that perception is based on the attributes describing the results of symptomatic tests, which describe the actual level of critical illness (eg. ECG signal). Let the administration of the Z drug be a factor interfering with perception. Let K1 and K2 are classifiers that have been constructed based on the training data to predict the A, B, C or D state, without the interfering factor and with its presence, respectively. If the K1 classifier assigns the particular patient P1 to state B, it turns out that the K2 classifier may classify P1 to each state: A, B, C or D.

This is so, because the distortion of perception causes the misrepresentation of patient’s image, which sometimes is perceived as if he/she had more advanced disease than it really is, and sometimes as if he/she had less advanced disease. At the same time, the misstated picture does not concern the real severity of the disease, but the current degree of threat to life, which this disease causes. Thus, if one could predict what kind of change in the disease perception the Z drug will trigger for a given patient, it
could be used to support the acute treatment (e.g. on the way to the hospital, awaiting surgery and so on).

Note, however, that the standard method of a classifier construction is not useful for predicting changes in the perception caused by a disturbing factor, because there is no possibility to construct a decision attribute that represents the changes of the perception of the disease for a given patient. This would require the experiments involving patients who had to be treated both without and with the use of the additional drug. Such a situation appears to be technically difficult and unethical, because it can generate a risk of less effective treatment (treatment extended in time).

The former work on this issue concerned the influence of two methods of surgical treatment of the throat and larynx cancers on the survival of patients ([2]). This work led to the discovery of a data mining method, which calculates the so-called cross-decision rules in medical data. Each cross-decision rule, for the group of patients described by its predecessor, gives the consequent likelihood of success of the treatment (patient survival) for both studied treatments. However, the method is based on the symbolic data attributes. This limitation comes from the need to count the specific reducts with regard to the complex decision, which in [2] are calculated for symbolic data. However, in many applications there is a need to analyze data with numerical attributes or mixed ones (symbolic and numeric). We therefore need rules whose predecessors can have ranges of values rather than specific attribute values.

Therefore, in this paper we propose a method based on the so-called tree of the impact, which is a binary tree. The impact tree is constructed so that the leaves contain descriptions of patient groups (patterns) whose perception of the disease changes in a similar way after the application of a disturbing factor. In some leaves a large change in perception is observed, while in some others the small one. Some leaves are characterized by a beneficial change, while certain other by an adverse one. With the patterns assigned to the leaves, the perception changes of the test objects can be predicted. Therefore, each leaf of the tree provides a single cross-decision rule, which in this work will be called a cross-decision rule of perception changes. The rules could be suggested as a solution to the problem of predicting changes in perception. For this purpose, the consequent of the rule would bring the information about the perception both without and with the disturbing factor.

2 Medical Background

Our considerations apply to the patients with stable coronary heart disease. The disease is characterized by reduced blood supply to the heart caused by atherosclerosis. The atherosclerosis is usually present in blood vessels even when their lumens appear normal in angiography. The CHD touches people all over the world and is one of the leading cause of deaths ([7]). Treatment may include medication or invasive revascularization. Treatment is aimed at reducing or eliminating symptoms and reducing the risk of a heart attack. The standard pharmacotherapy includes such classes of medications as: beta blockers, nitrates, calcium channel blockers/calcium antagonists or ACE inhibitors.

Recently a heated discussion on the inflammatory theory underlying the coronary artery disease is conducted ([4],[6]). Inflammatory mediators are involved in the de-
velopment, progression and destabilization of atherosclerotic plaques ([1]). Some of the strongest pro-inflammatory mediators are leukotrienes. Increased generation of leukotrienes is observed during acute myocardial ischemia, coronary angiography or angioplasty ([17]).

In this context, the studies on the use of anti-inflammatory drugs in CHD were conducted (eg. [13], [16]). In [16] the effect of pharmacological inhibition of leukotriene production on the electrical activity of the heart was assessed using an inhibitor of 5-lipoxygenase (zileuton). It has been shown that pharmacological inhibition of leukotriene biosynthesis in patients with stable CHD subjected to intracoronary interventions causes: reduction of the 24 hours average heart rate, increase in parameters of rhythm variability, and an improvement in the conduction of electrical impulses in the conduction system during intracoronary procedures. No effects on either arrhythmias or ECG patterns of ischemia were noted.

In view of the above results, it would be advantageous to indicate patients with beneficial effects of anti-inflammatory therapy. Commonly used methods based on global statistics such as the average or standard deviation, does not allow for selection of individual patients. Therefore, it is necessary to develop additional methods for evaluating the effect of treatment in individual patients. So we propose the use of a novelty data mining method using the tree measuring the impact of the factor interfering the perception. As a disturbing factor we use here an inflammatory drug - zileuton.

### 3 Tree Measuring the Impact of the Factor Interfering the Perception

The proposed cross-decision rules of perception changes are the kind of decision rules. A decision rule takes the form of implication, in which the left side consists of presumptions (conditions) expressed as a logical formula, and the right side contains a conclusion (thesis). In classical decision rules, the thesis identifies the decision class, while in the cross-decision rules it is a description of groups of objects from different decision classes.

For the problem of the coronary disease, the descriptors represent the expected value of the number of stenoses for different types of treatment. The Formula 1 presents an exemplary cross-decision rule of perception changes:

$$a = v_1 \land b = v_2 \Rightarrow \begin{cases} E(S \text{ after } A) = x_1 \\ E(S \text{ after } B) = x_2 \end{cases}$$

where $a, b \in A$ (the set of attributes), $E$ designates expected value, $S$ - the number of significantly narrowed coronary arteries and $A, B$ - therapy with placebo and with the disturbing factor.

For example, let the pattern $a = 1$ and $b > 2$ indicate a stable condition in ECG (without the danger). Then the cross-decision rule of perception changes can be as in Formula 2:

$$a = 1 \land b > 2 \Rightarrow \begin{cases} E(S \text{ after } P) = 0 \\ E(S \text{ after } Z) = 2 \end{cases}$$
where $P$ designates a treatment with placebo and $Z$ - with the disturbing factor (zileuton). Rule 2 means that the treatment causes a misrepresentation of the real state, determined by coronary angiography (number of stenoses). Despite two significantly narrowed coronary arteries, patients receiving zileuton have an ECG indistinguishable from placebo-treated patients without vascular changes.

The cross-decision rules are therefore a way of the knowledge presentation in an intelligible form that facilitates its interpretation. For the case of the CHD, they can be used to decide on the continuation of pharmacotherapy in a designated group of patients.

In order to generate the cross-decision rules of perception changes, we will use the impact tree. It is a binary tree constructed by the greedy algorithm in a top-down recursive divide-and-conquer manner. It takes a subset of data as an input and evaluates all possible splits. The best split is chosen to partition the data into two subsets (divide-and-conquer) and the method is called recursively. The algorithm stops when the stop conditions are met.

### 3.1 Construction of the Tree of Impact

As a criteria for selecting the best split, we propose a measure based on the distance between the groups. The measure is calculated using probability theory and statistical techniques, such as expected value, well-known from the literature [9],[14, 15].

Suppose that a set of objects includes two groups of subjects. One group was treated with a disturbing factor and the other not (patients received placebo). We are interested in the behavior of a certain characteristic in both groups. Its assessment is carried out according to the following steps:

1. Determination of the probability distributions of chosen feature in both groups, designated as $A$ and $B$
2. Definition of a variable representing the difference of the feature between the two groups: $X = |A - B|
3. Denotation of the distribution of $X$ variable
4. Determination of the expected value of $X$.

The expected value of the difference makes it possible to quantify the variation of the characteristic in both groups. Such a measure used to construct an impact tree, allows the assessment of the degree of influence of disturbing factor on the behavior of objects.

For the concept of CHD, the chosen characteristic may be the number of significantly narrowed coronary arteries which accepts four values: 0, 1, 2 and 3. Then, the distributions of the characteristic in both groups are presented in Tables 1 and 2.

The distribution of $X$ variable (difference of the characteristics between groups) is presented in Table 3, where the probability $p_i$, for $i = 0, 1, ..3$ is calculated using the
Table 1. Distribution of the chosen characteristic in placebo-treated group

<table>
<thead>
<tr>
<th>A</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>P(A)</td>
<td>a_0</td>
<td>a_1</td>
<td>a_2</td>
<td>a_3</td>
</tr>
</tbody>
</table>

Table 2. Distribution of the chosen characteristic in group subjected to a disturbing factor

<table>
<thead>
<tr>
<th>B</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>P(B)</td>
<td>b_0</td>
<td>b_1</td>
<td>b_2</td>
<td>b_3</td>
</tr>
</tbody>
</table>

Formulas in equations 3-6.

\[ p_0 = P(X = 0) = a_0 b_0 + a_1 b_1 + a_2 b_2 + a_3 b_3 \]  
\[ p_1 = P(X = 1) = a_0 b_1 + a_1 b_0 + a_2 b_3 + a_3 b_2 + a_2 b_1 + a_3 b_2 \]  
\[ p_2 = P(X = 2) = a_0 b_2 + a_1 b_3 + a_2 b_0 + a_3 b_1 \]  
\[ p_3 = P(X = 3) = a_0 b_3 + a_3 b_0 \]  

Table 3. Distribution of X variable

<table>
<thead>
<tr>
<th>X</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(X)</td>
<td>p_0</td>
<td>p_1</td>
<td>p_2</td>
<td>p_3</td>
</tr>
</tbody>
</table>

The expected value of \( X \) is calculated afterwards according to Formula 7.

\[ E(X) = 0 * p_0 + 1 * p_1 + 2 * p_2 + 3 * p_3 \]  

The stop condition of a tree construction is satisfied when the expected value of \( X \) variable exceeds a certain preset threshold \( th \). In addition, the divisions can be completed also in a situation where the number of objects in a given node falls below a certain level. The value of \( th \) was set to 1.75 for the impact tree in the figure 1.

The idea is to separate the groups of patients, with a large change in perception from those with little change. So the quality of the cut is defined by the Formula 8:

\[ Q = max|E(X_{left}) - E(X_{right})| \]  

The construction of the impact tree proceeds according to the algorithm 1. In a particular node of the impact tree, we determine the effect of a disturbing factor on the basis of the relationship between the expected number of stenoses in the untreated group and that value in treated one. The expected impact of the disruption is calculated by the Formula 9:
Algorithm 1: Construction of impact tree

**Step 1** Sort the values of the numerical attribute $a$

**Step 2** Browsing the values of $a$ attribute from the smallest to the largest for each appearing cut $c$ designate Euclidean distance between the expected values of $X$ feature

**Step 3** Select division among the possible divisions, such that $|E(X_{left}) - E(X_{right})| = \max$

**Step 4** Split the table $DT$ into two subtables $DT(T_p)$ and $DT(\neg T_p)$ such that $DT(T_p)$ includes objects matching the pattern $T_p$, and $DT(\neg T_p)$ includes objects matching the pattern $\neg T_p$.

**Step 5** IF the tables $DT(T_p)$ and $DT(\neg T_p)$ meet the stop conditions, then terminate tree construction

else repeat 1-4 for all tables which do not meet the stop condition

$$\delta = E(B) - E(A)$$

The factor interfering the perception may be deemed as beneficial, if the value of $\delta$ is greater than or equal to some assumed value $x$. When the $\delta$ value falls below a certain value $y$, than the factor can be regarded as an adversely affecting agent. In the case of CHD, we set the value of $x$ on 1.5, and $y$ on -1.5.

Due to the necessity of sorting attributes values done in time $O(n \cdot \log n)$, the computational complexity of the algorithm 1 is of the order $O(n \cdot m \cdot \log n)$.

4 Experiments and Results

The experiments have been conducted on dataset obtained from the Second Department of Internal Medicine, Collegium Medicum, Jagiellonian University, Krakow, Poland. The baseline and angiographic characteristics of the studied subjects are given in Tables 4 and 5. No significant differences with respect to age, gender or the angiographic characteristics were found between the study groups.

<table>
<thead>
<tr>
<th>Table 4. Baseline characteristics of the studied groups</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Placebo(n=33)</strong></td>
</tr>
<tr>
<td>Age average</td>
</tr>
<tr>
<td>Female/Male</td>
</tr>
</tbody>
</table>

Using the proposed methodology we achieved the tree of impact as shown in Fig. 1. The tree contains six leaves. An example of the beneficial effect of the additional therapy (zileuton) in CHD represents the rule in one of the leaves shown in Formula
Fig. 1. An example of the impact tree for CHD

10. The parameter $AVG_{ST\_DOWN3}$ signifies a daily average of the maximum ST segment depression in particular hours [mV]. $FIRST_{VLF}$ - heart rate variability (HRV) power spectrum [$ms^2$] in the range of very low frequencies ($0.0033 - 0.04Hz$) in the first hour of the ECG Holter recording and $AVG_{QT2\_AVG}$ - a daily average of the QT interval duration average in particular hours [$ms$]. The pattern based on Holter ECG parameters is common to untreated patients without coronary stenoses and the patients with significant stenoses treated with zileuton. In terms of the ECG features, they are indistinguishable. Therefore, additional treatment modifies the ECG, such as found in untreated patients without stenosis.
Table 5. Angiographic characteristics of the study groups

<table>
<thead>
<tr>
<th>Placebo (n=33)</th>
<th>Zileuton (n=37)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-vessel coronary disease</td>
<td>15 (45.5%)</td>
</tr>
<tr>
<td>1-vessel coronary disease</td>
<td>5 (15.2%)</td>
</tr>
<tr>
<td>2-vessel coronary disease</td>
<td>6 (18.2%)</td>
</tr>
<tr>
<td>3-vessel coronary disease</td>
<td>7 (21.2%)</td>
</tr>
</tbody>
</table>

\[
AVG_{ST\_DOWN} N3 < -0.06 \land FIRST_{VLF} \geq 373 \\
\land AVG_{QT2\_AVG} \geq 464.2 \Rightarrow E(S \text{ after } P) = 0.29 \\
E(S \text{ after } Z) = 2.33
\]

(10)

The opposite case represents, for example, the cross-decision rule given by the Formula 11. The patients with such parameters do not benefit from the additional treatment. Despite the additional treatment, their ECG is the same as in untreated patients with significantly narrowed vessels.

\[
AVG_{ST\_DOWN} N3 < -0.06 \land FIRST_{VLF} < 373 \\
\Rightarrow E(S \text{ after } P) = 2.17 \\
E(S \text{ after } Z) = 0.2
\]

(11)

For patients who match the exemplary pattern from the rule presented in Formula 12, using our method, we have no basis for determining whether the anti-inflammatory treatment is beneficial. In this rule, \(AVG_{QT1\_STD}\) signifies a daily average of the QT interval standard deviation in particular hours \([ms]\).

\[
AVG_{ST\_DOWN} N3 < -0.06 \land FIRST_{VLF} \geq 373 \\
\land AVG_{QT2\_AVG} < 464.2 \land AVG_{QT1\_STD} \geq 26.1 \Rightarrow E(S \text{ after } P) = 0.25 \\
E(S \text{ after } Z) = 1.5
\]

(12)

5 Conclusions

We discussed the method for describing the effect of a disturbing factor on perception. We considered the influence of anti-inflammatory therapy on perception of narrowing in the blood vessels modeled by the tree of impact. The presented solution may be used in the case of CHD for maintaining momentary stability in the ambulance or in the preoperative period after myocardial infarction.

Our method chooses ECG parameters which are of great clinical importance, such as ST-segment depression associated with myocardial ischemia, very low frequency (VLF) oscillations in the power spectra of HRV or duration and dispersion of QT interval.
Interestingly, the method selects the parameters of ST-segment, on which anti-inflammatory treatment had no effect according to previous studies [13],[16]. The reason may lie in the use of global statistics, which do not distinguish between individual patients.

In the literature, HRV turned out to be a predictive factor of a cardiac death related to myocardial infarction and diabetic angiopathy ([5],[8]). Suppressed HRV, especially within its high frequency component, was found predictive for myocardial infarction or unstable coronary artery disease. While the prolongation of QT interval can predispose to a potentially fatal ventricular arrhythmia [11]. The factor interfering perception could be administered continuously in the case of proven efficacy for specific cases described by the proposed rules.

The novelty of our method is the use of well-known statistics as an innovative measure of a cut quality and the application of the cross-decision rules of perception changes to represent different behaviours of groups of patients.

However, this method is not free from disadvantages. One of them is the need to match a particular type of data, that is diversified within the concept (e.g. different number of stenoses). Another threat is a manual adjustment of the threshold values, which is always a risk of a misapplication. The weak point of the experiment is small data set.

In future we are going to continue the experiments concerning the considered medical problem, in purpose of extending the results of this paper. We also intent to strengthen the reliability of the method for its clinical use. We plan to develop a general approach to describing the influence of a disturbing factor on the perception.

Acknowledgement

This work was partially supported by two following grants of the Polish National Science Centre: DEC-2013/09/B/ST6/01568, DEC-2013/09/B/NZ5/00758, and also by the Centre for Innovation and Transfer of Natural Sciences and Engineering Knowledge of University of Rzeszów, Poland.

References

Dialogue in Hierarchical Learning of a Concept Using Prototypes and Counterexamples

Extended Abstract

Soma Dutta and Piotr Wasilewski
Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw, Poland
somadutta9@gmail.com
piotr@mimuw.edu.pl

While dealing with vague concepts often it puts us in fix to determine whether to a particular situation/case/state a particular concept applies or not. A human perceiver can determine some cases as the positive instances of the concept, and some as the negative instances of the same; but there always remain cases, which might have some similarities with some positive cases, and also have some similarities with some negative cases of the concept. So we propose to learn about the applicability of a concept to a particular situation using a notion of similarity of the situation with the available prototypes (positive instances) and counterexamples (negative instances) of the concept. Perceiving a vague concept, due to the inherent nature of vagueness, is subjective, and thus never can be exhausted by listing down all the positive and negative instances of the concept. Rather we may come to realize about the applicability, or non-applicability, or applicability to some extent, of a concept to a situation in a step-by-step hierarchical manner by initiating dialogue between a perceiver and the situation descriptor. Hence, the main key ingredients of this proposal are (i) prototypes and counterexamples of a concept, (ii) similarity based arguments in favour and against of applicability of a concept at a particular situation, and (iii) hierarchical learning of the concept through dialogues. Similarity based reasoning [3], hierarchical learning of concepts [1], dialogue in the context of approximation space [2] all are separately important directions of research. For our purpose, in this presentation we would concentrate on combining these aspects from a different angle.

In [4], a preliminary version of logic of prototypes and counterexamples has been set. To make this paper self-contained, we recapitulate the necessary definitions below.

We start with a set $S$ of finitely many situations, say $\{s_1, s_2, \ldots, s_n\}$, and $A$ of finitely many attributes $\{a_1, a_2, \ldots, a_m\}$. Each $s_i$, $i = 1, 2, \ldots, n$, is considered to be a function $s_i : A \rightarrow [0, 1]$. Let the consolidated data of each situation is stored in the form of a set $\langle s_i(a_1), s_i(a_2), \ldots, s_i(a_m) \rangle : s_i \in S$, which is a subset of $[0, 1]^m$. Let $W \subseteq [0, 1]^m$ and $\{\langle s_i(a_1), s_i(a_2), \ldots, s_i(a_m) \rangle : s_i \in S\} \subseteq W$. Each member of $W$ may be called a world. We now consider a fuzzy approximation space $\langle W, Sim \rangle$, where $Sim$ is a fuzzy similarity relation between worlds of $W$. That is, $Sim : W \times W \rightarrow [0, 1]$, and we assume $Sim$ to satisfy the following properties.
(i) $Sim(\omega, \omega) = 1$ (reflexivity)
(ii) $Sim(\omega, \omega') = Sim(\omega', \omega)$ (symmetry)
(iii) $Sim(\omega, \omega') \cdot Sim(\omega', \omega'') \leq Sim(\omega, \omega'')$ (transitivity).

Following [3], the fuzzy approximation space $\langle W, Sim \rangle$ is based on the unit
interval $[0, 1]$ endowed with a t-norm $\ast$ and a S-implication operation $\rightarrow$. We now propose to represent any (vague) concept $\alpha$ by a pair $(\alpha^+, \alpha^-)$ consisting of the positive instances (prototypes) and negative instances (counterexamples) of $\alpha$ respectively, where $\alpha^+, \alpha^- \subseteq W$ and $\alpha^+ \cap \alpha^- = \phi$.

**Definition 1** [4]. Given the fuzzy approximation space $\langle W, \text{Sim} \rangle$, and a concept $\alpha$ represented by $(\alpha^+, \alpha^-)$, the degree to which $\alpha$ applies to a world $\omega \in W$, denoted by $\text{gr}(\omega \models \alpha)$, is given by:

\[
\text{gr}(\omega \models \alpha) = \begin{cases} 
0, & \text{if } \omega \in \alpha^-, \\
\text{Sim}^\alpha(\omega) & \text{if } \omega \in \alpha^+.
\end{cases}
\]

The fuzzy upper approximations $\overline{\text{Sim}^\alpha}$ and $\overline{\text{Sim}^\alpha}$ are defined following the Definition proposed in [3], i.e., $\overline{\text{Sim}^\alpha}(\omega) = \sup_{u \in W} \text{Sim}(\omega, u) \ast \alpha^+(u) = \sup_{u \in \alpha^+} \text{Sim}(\omega, u)$. Similar is the case for $\overline{\text{Sim}^\alpha}$. $\sim$ is considered to be the standard complementation operation defined as $\sim a = 1 - a$.

Let us call $\overline{\text{Sim}^\alpha}(\omega) = D_{af}(\omega, \alpha)$, the degree of arguments in favour of $\omega$ qualifies $\alpha$ and $\overline{\text{Sim}^\alpha}(\omega) = D_{af}(\omega, \alpha)$, the degree of arguments against $\omega$ qualifies $\alpha$. So, given a concept $\alpha$ and world $\omega \notin \alpha^+, \alpha^-$, $\text{gr}(\omega \models \alpha) = D_{af}(\omega, \alpha) \ast -D_{af}(\omega, \alpha)$.

Let us now pose the issue of the research in disguise of a practical need. Let we have a clinical record of $n$ number of patients’ details with respect to some $m$ number of parameters/attributes. These parameters might be some objective values of some clinical tests, called signs, or some subjective features experienced by the patients, called symptoms. With respect to the state of each patient, the values corresponding to all these parameters are converted, by some mean, to the values over a common scale, say $[0, 1]$. That is, if an $m$-tuple $\langle x_1, x_2, \ldots, x_m \rangle$ from $[0, 1]^m$ represents the rates of the $m$ parameters corresponding to a patient, then we say $\langle x_1, x_2, \ldots, x_m \rangle$ describes the state of a patient. Based on the rates assigned to all the parameters by each patient, i.e. a $m$-tuple of values $\langle x_1, x_2, \ldots, x_m \rangle$ from $[0, 1]^m$, which cases representing the states of the patients are how much similar or dissimilar may be anticipated. Now, one task is to make a tentative diagnosis about a patient whose measurement concerning the $m$-tuple of parameters appears to be new with respect to the database of the $n$ patients. Now with the above set-up, developed in [4], we may compute $\text{gr}(\omega \models d)$, the degree of applicability of a disease $d$ for the newly appeared situation, say world $\omega$. The value $\sim$, $\text{gr}(\omega \models d)$, for different diseases, may help to make a hypothetical assumption regarding the plausible disease. For being more certain about the diagnosis, it is quite natural to enquire about some more factors/attributes. So the dialogue would have a role to play here. In order to incorporate dialogue in the previous set-up, below we would present the above mentioned theory in a broader framework.

Let us first fix the domain of the (vague) concepts of our concern. Let $A$ be the set of all attributes (finitely many) required to understand all the concepts over the fixed domain. At time $t_0$, with respect to a set of attributes $A_{t_0} \subseteq A$ we have a set of finitely many situations, say $S_{t_0}$, at hand such that which situation is characterized by which concept is known to us. That is, given a situation $s$ from $S_{t_0}$, $s$ is characterized as a positive instance or negative instance of some of the concepts $c$ over the domain of concern. So, we say $S_{t_0}$, a set of situations, is characterized by the set of attributes
\( A_{t_0} \) at time \( t_0 \). Let \( S_{t_0} = \{s_{1t_0}, s_{2t_0}, \ldots, s_{nt_0}\} \) and \( A_{t_0} = \{a_1, a_2, \ldots, a_m\} \). We say a database at time \( t_0 \) with respect to the set of situations \( S_{t_0} \), denoted as \( D_{S_{t_0}} \), is the set of all tuples of values for the attributes of \( A_{t_0} \) for each situations of \( S_{t_0} \).

That is, for each \( a_i(s_{jt_0}) \in [0,1], i \in \{1, 2, \ldots, m\} \) and \( j \in \{1, 2, \ldots, n\} \), \( D_{S_{t_0}} = \{\langle a_1(s_{jt_0}), \ldots, a_m(s_{jt_0}) \rangle : s_{jt_0} \in S_{t_0}\} \subseteq [0,1]^m \). According to rough set literature, \( D_{S_{t_0}} \) is basically an information system. We assume that for each database \( D_{S_{t_0}} \) there is a database manager, may be called decision maker, \( dm(S_{t_0}) \).

**Definition 2:** A dialogue base at some time point \( t_k \) is a tuple \((G_1, G_2, \ldots, G_r, R_i, R_e)\) such that each \( G_j = \langle A_{G_j}, A_j, R_i \rangle \) constitutes of a set of agents \( A_{G_j} \), a set of attributes \( A_j \), and an accessibility relation \( R_i \) among the agents. \( R_i \) stands for internal accessibility relation among the agents of each group \( G_j \). Each \( A_{G_j} \supseteq S_{t_k}^j \cup \{dm_j\} \) for some time point \( t_k \), where \( S_{t_k}^j \) is the a set of situations characterized by \( A_j \). That is each \( A_{G_j} \) contains a set of situations \( S_{t_k}^j \) and a database manager \( dm_j \) corresponding to \( S_{t_k}^j \). For each \( s \in S_{t_k}^j \), \( R_e(s, dm_j) \) holds, and \( R_i \) is symmetric. The relation \( R_e \) is a reflexive, symmetric relation and it stands for external accessibility relation between different database managers. That is, for some \( j, l, R_e(dm_j, dm_l) \) holds.

Intuitively each \( G_j \) of the dialogue base contains a set of nodes and relation \( R_i \) among the nodes. These nodes constitute the set \( A_{G_j} \). Some of the nodes represent those situations which are characterized by \( A_j \), the set of attributes of \( G_j \). \( dm_j \) is a node designated as database manager. \( S_{t_k}^j \), the set of situations characterized by \( A_j \), generates a database \( D_{S_{t_k}^j} \) of tuple of values for each attribute of \( A_j \) corresponding to each situation of \( S_{t_k}^j \). \( D_{S_{t_k}^j} \subseteq W_j \), and hence is embedded in the approximation space \((W_j, Sim_{A_j})\). In each \( G_j \) the \( dm_j \) has access to the other nodes. Through dialogue it is expected that \( dm_j \) would enquire a particular situation (i.e. node) for information, and the particular situation would provide the information corresponding to the query. So, \( R_i \) has to be symmetric as both database manager and the situation descriptor should have access to make the communication. The external accessibility relation \( R_e \) allows accessing two database managers. A database manager can access her own information, and if \( dm_j \) can access \( dm_l \), then the reverse also holds. So, \( R_e \) is reflexive and symmetric.

Summarizing the whole, we can say that each \( A_{G_j} \) of \( G_j \) is a set of nodes some of which are specific cases, already characterized by the set of attributes \( A_j \) of \( G_j \), at some time point. \( dm_j \) can be considered as a dummy node which can access any other node. The rest of nodes can be any new case/situation appearing at some further point of time. That is why \( A_{G_j} \supseteq S_{t_k}^j \cup \{dm_j\} \). On the other hand, the detailed information about the set of situations \( S_{t_k}^j \), which are characterized by \( A_j \), are available in the corresponding database (or information system) \( D_{S_{t_k}^j} \). The information system or database \( D_{S_{t_k}^j} \) is also open to handle new information corresponding to new cases of \( A_{G_j} \). That is why, \( D_{S_{t_k}^j} \subseteq W_j \), where \((W_j, Sim_{A_j})\) is a fuzzy approximation space based on the attributes \( A_j \). The following picture is a model of what we are trying to formalize through the notion of dialogue base and approximation spaces containing.
different databases characterized by different sets of attributes.

Fig. 1. Internal and external dialogues among the granules of a dialogue base and corresponding outcomes generated in the respective approximation spaces

Now given the prelude of the practical need, what do we expect from a dialogue? A dialogue at the first time point $t_0$, denoted as $diag_{t_0}$, would consist of two rounds $r_1$ and $r_2$. At round $r_1$ the database manager may ask the situation $s$ to provide the values for a set of attributes $\langle a_1, \ldots, a_m \rangle$. At the round $r_2$ the situation descriptor answers the query with the tuple $\langle a_1 = v_1, \ldots, a_m = v_m \rangle$, or in other words simply a tuple of values from $[0, 1]^m$. So, as an output of a complete dialogue at time $t_0$ between a situation $s$ and the corresponding database manager $dm_{t_0}$ we expect to receive a tuple of values from $[0, 1]^m$. Though we are going to combine the two rounds in a single complete dialogue, there is a difference in the nature of the two rounds. The dialogue in $r_1$ throws a question, and the dialogue in $r_2$ provides an answer. So, the dialogue somehow moves the communication from the first agent’s approximation space to the second agent’s approximation space. So, the definition of dialogue is proposed as follows.

**Definition 3**: Given a dialogue base $(G_1, G_2, \ldots, G_r, R_i, R_e)$ at time $t_0$, a dialogue between two agents $ag_1, ag_2$, denoted as $diag_{t_0}(ag_1, ag_2)$, is defined as follows. 
(i) $diag_{t_0}(ag_1, ag_2) = \omega \in W_i$ of the approximation space $(W_i, Sim_{At_i})$, if $R_i(\omega, [0, 1]^m)$ holds for $ag_1, ag_2$ in $Ag_i$ of the group $(Ag_i, At_i, R_i)$, and $W_i \supseteq D_{St_0}$ for
Step 4 At time $t_0$.

Step 5 Now in the fuzzy approximation space ($G_{S_{t_0}}$).

Step 3 Now we start with a dialogue base ($G_{S_{t_0}}$, Sim, $\mathcal{A}_{S_{t_0}}$) at time $t_0$. For each $i = 1, 2, \ldots, r$, $G_{S_{t_0}} = (\mathcal{G}_{S_{t_0}}, \mathcal{A}_{S_{t_0}}, R_i, R_e)$ and $\mathcal{A}_{S_{t_0}} \supseteq S_{t_0} \cup \{d_{m_{S_{t_0}}}\}$, where $S_{t_0}$ is the set of situations characterized by $\mathcal{A}_{S_{t_0}}$. Each $S_{t_0}$ is embedded in an approximation space ($W_{S_{t_0}}$, Sim, $\mathcal{A}_{S_{t_0}}$) through its database $D_{S_{t_0}}$. To mark the time point $t_0$ corresponding to each component of a dialogue base we have used suffixes like $i_0$. But every $i_0$ must coincide with some $S_j$ of situations, about which we have discussed at Step 2, as either of the groups ($G_{S_{t_0}}$, 1 $\leq i \leq r$) of the dialogue base constitutes of a set of agents and a set of attributes taken from $\mathcal{A}$.

Step 4 At time $t_0$, given a situation $s \in \mathcal{A}_{S_{t_0}}$ - $\{d_{m_{S_{t_0}}}\}$ a dialogue is initiated as $\text{diag}(s, d_{m_{S_{t_0}}})$. The output of the dialogue would provide a tuple of values from $W_{S_{t_0}}$. Let us assume that $\text{diag}(s, d_{m_{S_{t_0}}}) = \omega \in W_{S_{t_0}}$.

Step 5 Now in the fuzzy approximation space ($W_{S_{t_0}}$, Sim, $\mathcal{A}_{S_{t_0}}$), based on the development made in [4], we can compute $gr(\omega |= e c)$ (see Definition 1) for a concept belonging
to $C$. Based on some (significantly high) value $gr(\omega \models c)$ for some $c$, we can make a hypothesis for the ‘applicability of the concept $c$ at the world $\omega (= s(t_0))$’, the world assigned by situation $s$ at time $t_0$.

Step 6 To be more certain regarding the decision, at time $t_1 (> t_0)$ the second dialogue may be initiated as $diag_{t_1}(s, dm_{it_1})$, where $dm_{it_1}$ is the same database manager $dm_{it_0}$ at the next time point $t_1$, as the old dialogue base changes to $(G_{it_1}, G_{2t_1}, \ldots, G_{rt_1}, R_i, R_e)$ considering the new time point. The new $G_{it_1}$ contains all the agents of $G_{it_0}$ and preserves the same relation $R_i$ among the agents of $G_{it_0}$. It differs in the set of attributes $A_{it_1}$ where $A_{it_1} \supseteq A_{it_0}$. As, $A_{it_0} \subseteq A_{it_1}$, $S_{it_0} \subseteq S_{it_1}$. We take $G_{it_1} = G_{it_0} \cup S_{it_1}$, where $S_{it_1}$ is the set of situations characterized by $A_{it_1}$. So, $diag_{it_1}(s, dm_{it_1})$ would now provide a new world $\omega'$ from the approximation space $(W_{it_1}, Sim_{A_{it_1}})$. $W_{it_0}$ is embedded in $W_{it_1}$ in the sense that $W_{it_1} \subseteq W_{it_0} \times [0, 1]^{A_{it_1} \setminus A_{it_0}}$, and for each $u = \langle u_1, \ldots, u_{|A_{it_0}|} \rangle \in W_{it_0}$, there is $u' \in W_{it_1}$ such that $u' = \langle u_1, \ldots, u_{|A_{it_1}|}, 0, 0, \ldots, 0 \rangle$ having entry ‘0’ for the rest of the $|A_{it_1}| - |A_{it_0}|$ components. When a dialogue at time $t_i$ moves to a new approximation space from its previous approximation space at time $t_{i-1}$ we call the dialogue proceeds.

Step 7 As all the subsets of the whole attribute set $A$ is considered, the set of attributes $A_{it_1} \subseteq A$ must merge with some $A_j$ considered in the beginning. So, there is already a set of situations and corresponding database embedded in the approximation space $(W_{jt_1}, Sim_{A_j}) = (W_j, Sim_{A_j})$, and the dialogue at time $t_j$ moves into the new approximation space with respect to $A_j$. So, with respect to the approximation space $(W_j, A_j)$ we can compute $gr(\omega' \models c)$.

Step 8 Now, if $gr(\omega \models c) < gr(\omega' \models c)$ where $s(t_0) = \omega$ and $s(t_1) = \omega'$, then we may consider the situation $s$ to be ascribed as an instance of $c$.

Example: Let we have a clinical database of a set of situations, $S = \{s_1, s_2, s_3, \ldots, s_9\}$ with respect to a set $A$ of attributes, consisting of temperature, blood-pressure, blood-tests, ecg, headache, sneezing, convulsion, vomiting, skin-rash, dizziness, stomach-upset, stomach pain. Sequentially let us call these attributes as $a_1, a_2, a_3, \ldots, a_{12}$. As $a_1, \ldots, a_4$ are determined by some objective values of some tests these are called signs; the rest are symptoms, determined by some subjective values as experienced by particular patients. Let $C_B = \{Fev, Allergy, Stomachinf, HBP, LBP, Vertigo, Unconsciousness\}$, and $C$ is the union of $C_B$ and $\{Fevc, Fevc, Stroke, Food-poisoning, Viralinf, Pepticulcer\}$. The relations among the dependent and the independent concepts of $C$ are as follows.
Let us start at time $t_0$ with the dialogue base $(G_t, R_c, R_s)$. For simplicity we have considered only one group of agents having internal relation $R_i$ and external relation $R_c$. Let $S_{t_0} = S$, and at time $t_0$ the set of situations $S_{t_0}$ is characterized by $A = A_{t_0}$. The database at time $t_0$, denoted as $D_{S_{t_0}}$, is basically the set \{$(s_1(a_1), s_1(a_2), \ldots, s_1(a_{12}))$ : $s_i \in S_{t_0}$\}. Now $G_{t_0} \supseteq S_{t_0} \cup dm_{t_0}$, where $dm_{t_0}$ is a dummy agent representing the database manager for $D_{S_{t_0}}$, and $D_{S_{t_0}} \subseteq \mathbb{W}_{t_0} \subseteq [0,1]^{12}$. For each $s_i \in S_{t_0}$, $R_i(dm_{t_0}, s_i)$ holds, and $R_i$ is symmetric. Now in appearance of a new situation $s_{10}$, the outcome of the dialogue between $dm_{t_0}$ and $s_{10}$ at the round

**Table 1. Patients data table**

<table>
<thead>
<tr>
<th></th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_5$</th>
<th>$a_6$</th>
<th>$a_7$</th>
<th>$a_8$</th>
<th>$a_9$</th>
<th>$a_{10}$</th>
<th>$a_{11}$</th>
<th>$a_{12}$</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
<th>$d_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>.8</td>
<td>.5</td>
<td>.5</td>
<td>.0</td>
<td>.7</td>
<td>.7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td>.5</td>
<td>.8</td>
<td>.7</td>
<td>.7</td>
<td>0</td>
<td>0</td>
<td>.3</td>
<td>.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>$s_3$</td>
<td>.9</td>
<td>.5</td>
<td>.7</td>
<td>.0</td>
<td>.8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>$s_4$</td>
<td>.4</td>
<td>.3</td>
<td>.6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.0</td>
<td>.5</td>
<td>0</td>
<td>.0</td>
<td>.7</td>
<td>.6</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>$s_5$</td>
<td>.7</td>
<td>.1</td>
<td>.4</td>
<td>.7</td>
<td>0</td>
<td>.3</td>
<td>0</td>
<td>0</td>
<td>.7</td>
<td>0</td>
<td>0</td>
<td>.9</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>$s_6$</td>
<td>.1</td>
<td>.2</td>
<td>.8</td>
<td>.0</td>
<td>.3</td>
<td>0</td>
<td>.0</td>
<td>.5</td>
<td>.7</td>
<td>0</td>
<td>.7</td>
<td>.7</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>$s_7$</td>
<td>.9</td>
<td>.5</td>
<td>.8</td>
<td>.0</td>
<td>.8</td>
<td>.0</td>
<td>.0</td>
<td>.3</td>
<td>.2</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>$s_8$</td>
<td>.3</td>
<td>.5</td>
<td>.3</td>
<td>.7</td>
<td>.0</td>
<td>.7</td>
<td>0</td>
<td>.0</td>
<td>.8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>$s_9$</td>
<td>.6</td>
<td>.7</td>
<td>.7</td>
<td>.5</td>
<td>.5</td>
<td>0</td>
<td>0</td>
<td>.9</td>
<td>0</td>
<td>.2</td>
<td>.9</td>
<td>1</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Let a new situation $s_{10}$, with the tuple $\langle 5, 5, 5, 5, 7, 0, 8, 5, 7, 8, 5, 0 \rangle$ of values corresponding to the respective attributes, appear. The task is to make a diagnosis for $s_{10}$. Now we pose the above problem in the framework of the present proposal in the following way.

For each $s_i \in S_{t_0}$, $R_i(dm_{t_0}, s_i)$ holds, and $R_i$ is symmetric. Now in appearance of a new situation $s_{10}$, the outcome of the dialogue between $dm_{t_0}$ and $s_{10}$ at the round...
$r_i$ is $\text{diag}^{r_i}_0(dm_{t_0}, s_{10}) = \langle a_1, a_2, \ldots, a_{12} \rangle$, and that of at the second round of the dialogue is $\text{diag}^{r_2}_0(s_{10}, dm_{t_0}) = \langle .5, .5, .5, .7, 0, .8, .5, .7, .8, .5, 0 \rangle$. Combining both the rounds we write $\text{diag}^{r_0}_0(s_{10}, dm_{t_0}) = \langle .5, .5, .5, .7, 0, .8, .5, .7, .8, .5, 0 \rangle = w$ (say).

Now based on the proposal presented in [4], with respect to the fuzzy approximation space $(W_{t_0}, \text{Sim}_{A_{t_0}})$ one can calculate $gr(w \models c)$ for some $c \in C$. Let us denote the degree to which $s_{10}$ qualifies $c$ at time $t_0$ as $gr(w \models^{t_0}_0 c)$. Let for $c = \text{Stroke}$, $gr(w \models^{t_0}_0 c) = \frac{1}{2}$. In order to be more certain the decision maker may need to ask the patient for some more tests. Let the new test, i.e., the attribute $a_{13}$ is $\text{MRI-scan}$ (Magnetic Resonance Imaging). So, the dialogue base at the next time point $t_1$ moves to $(G_{t_1}, R_c, R_i)$ where $G_{t_1} = G_{t_0} \cup S_{t_1} \supseteq S_{t_1} \cup \{dm_{t_1}\}, dm_{t_1}$ is the same as $dm_{t_0}$ at the next point of time, and $S_{t_1}$ is the set of situations characterized by $A_{t_1} = A_{t_0} \cup \{a_{13}\}$.

According to the definition of a dialogue base $R_i(dm_{t_1}, s_i)$ holds for each $s_i \in S_{t_1}$. Now the new dialogue at time $t_1$ would be $\text{diag}^{r_1}_{t_1}(dm_{t_1}, s_{10}) = \langle a_1, a_2, \ldots, a_{12}, a_{13} \rangle$, and let $\text{diag}^{r_2}_{t_1}(dm_{t_1}, s_{10}) = \langle s_{10}(a_1), s_{10}(a_2), \ldots, s_{10}(a_{12}), s_{10}(a_{13}) \rangle = w' \in W_{t_1}$, where $W_{t_1} \subseteq W_{t_0} \times [0, 1)$ such that for each $w = \langle x_1, x_2, \ldots, x_{12} \rangle \in W_{t_0}$ there is $\langle x_1, x_2, \ldots, x_{12}, 0 \rangle \in W_{t_1}$. So, now at time $t_1$ with respect to the fuzzy approximation space $(W_{t_1}, \text{Sim}_{A_{t_1}})$ one can calculate $gr(w' \models^{t_1}_c c)$, and based on $gr(w' \models^{t_1}_c c) \geq gr(w \models^{t_0}_0 c)$ or $gr(w' \models^{t_1}_c c) \leq gr(w \models^{t_0}_0 c)$ a decision regarding considering $s_{10}$ as a positive or negative case of Stroke may be taken through a hierarchical manner of learning.

Acknowledgement

Authors of this abstract are thankful to Professor Andrzej Skowron for his suggestion to introduce a framework of dialogue in the development of this work. This work has been carried out during the tenure of ERCIM Alain Bensoussan fellowship of the first author, and this joint work is partially supported by the Polish National Science Centre (NCN) grants DEC-2011/01/D/ST6/06981.

References

An Approach to Ambiguity Resolution for Ontology Population *

Natalia Garanina and Elena Sidorova
A.P. Ershov Institute of Informatics Systems, Lavrent’ev av., 6, Novosibirsk 630090, Russia
{garanina,lsidorova}@iis.nsk.su

Abstract. We study a problem of information retrieval and ambiguity resolution for ontology population. The process for retrieval of information in a form of a set of ontology instances is presented as a Scott information system. This representation proves termination of the process. The Scott information system is a basis for our approach to context-dependent ambiguity resolution. This system generates a multi-agent system in which agents resolve the ambiguity by computing the cardinality of their contexts.

1 Introduction

Ontological databases are currently widely used for storing information obtained from a great number of sources. To complete such ontologies, formalisms and methods that allow one to automate the process are developed.

We propose to consider the process of ontology population as work with information systems, which are concepts of the domain theory [9]. An information system is a universal model for knowledge (ontologies, thesauri, databases, etc.) organization systems. Information systems are “syntactic” representations of the Scott domains. They are simple, well-studied, and, in the context of ontology population, possess, in particular, useful properties of the entailment relation. In the framework of the algebraic approach to work with ontologies, theory of Formal Concept Analysis is usually applied, which makes it possible to enrich the ontology with new concepts [2]. The Scott domain theory was used for enriching ontologies with topological relations [4].

In this paper, we show that an ontology population system is a Scott information system, which gives an idea of certain general properties of the population process, in particular, a simple proof of termination of the population process and justification of the resolution of context-dependent semantic ambiguity. Features of information retrieval cause ontology population ambiguities. These ambiguities arise when several ontology instances are formed for the same input data. We use the information system of ontology population for resolving context-depending ambiguities. We can evaluate the cardinality of the instance contexts, i.e. how much an instance is related with the

* The research has been supported by Russian Foundation for Basic Research (grant 13-01-00643, 15-07-04144, 13-07-00422) and the People Programme (Marie Curie Actions) of the European Union’s Seventh Framework Programme FP7/2007-2013/ under REA grant agreement n PIRSES-GA-2011-294962-COMPUTAL.
other retrieved instances via the information contained in it. Thereby, the information system of ontology population generates a multi-agent system in which agents are ontology instances informationally connected. They are in conflict when there is the ontology ambiguity for them. Agents resolve the conflict by removing the less integrated agents with all their connections from the system. Hence the system is dynamic.

The works on multi-agent systems usually focus on the behavior of agents, methods of communication between agents, knowledge and belief of an agent about environment and other agents, etc [3, 10]. Works about conflict resolution process usually consider the process in terms of the behavior of the agent depending on its internal state, reasoning and argumentation methods etc. [7]. The dynamics of the agents connections is not a subject of these researches. On the other hand, there are studies on social networks, in which agents are connected by typed connections, but their weight is irrelevant [1].

The allied approach to ontology-driven disambiguation which uses a context of a given input text is suggested in [6]. Sequential sentence-by-sentence ambiguity resolution is performed. For every sentence, a set of semantic interpretation is constructed. Each interpretation is represented by a graph with vertices as ontology entities and edges as ontology relations. Every next sentence may increase scores of some interpretation of the previous ones. Hence the preference of alternative meaning of words is based on ontology relations. This approach has low accuracy because such on-the-fly choice of alternative takes into account the nearest context mainly. Our approach uses the maximal context available from an input text for choosing alternative. Connections between ontology objects (class and relation instances) correspond to informational dependance of the text. Since the approach does not require complete sentence parsing, this considerably simplify the text analysis for ontology population.

We propose a multi-agent algorithm for the conflict resolution in the generated multi-agent system of ontology instances. This algorithm is based on an idea of connectivity cardinality of instance agents. On a run of the algorithm agents compute their system weight corresponding to this connectivity. According to these weights, conflicts are resolved by a special agent. This agent adjusts the weight computing processes and constructing a conflict-free set of agents which is an optimal solution of a conflict resolution problem for a given system.

The rest of the paper is organized as follows. In Section 2, an approach to ontology population in the framework of information systems is discussed. The next Section 3, gives definitions for a multi-agent system for ambiguity resolution. Section 4 describes agents of our systems, their action protocols, and the main conflict resolution algorithm. In the concluding Section 5, directions of future researches are discussed.

2 Ontology Population and Scott Information Systems

Let we are given by an ontology of a subject domain, the ontology population rules, semantic and syntactic model for a sublanguage of the subject domain and a data format, and input data with information for population of the ontology. We consider an ontology of a subject domain which includes (1) a finite nonempty set of classes for concepts of the subject domain, (2) a finite set of named binary relations on the classes, (3) a finite set of attributes, and (4) a finite set of data types. Every class is defined by a tuple of
typed attributes. Every relation is defined by a couple of classes and a tuple of typed attributes. Information content of an ontology is a set of class instances and relation instances defined on these class instances.

Ontology population problem is to compute information content for a given ontology from given input data. A mapping of input data to the ontology has to be a one-to-one correspondence, i.e. different ontology instances correspond to different data fragments. Rules for ontology population and data processing determine ontology instances in input data, and evaluate their attributes. Determination of the information content includes the following stages. First, input data are matched to characters of attributes, classes, and relations of the ontology. These matchings are predefined and depend on an input data format. Initial ontology instances are formed using obtained ontology characters independently from each other. At this stage the instances are not determined completely, because information, which is necessary for evaluating their attributes and establishing relations, frequently places at isolated parts of input data. This information is usually stored in different ontology instances. At the second stage initial instances are formed consistently and completely (as much as possible), i.e. information from some instances could be used for evaluating other and for defining new ontology instances. Rules for ontology population and data processing define informationally connected instances. In paper [5] multiagent algorithm for the second stage is presented. In a process of instance determination, some ambiguities are possible when several ontology instances are formed for the same data fragment. These ambiguities can be caused by homonymy, references, etc. The problem of ambiguity resolution is solved at the third stage. This paper suggests an approach for the solving. At the forth stage ontology population itself is performed. Computed instances could be included into the (possibly nonempty) ontology informational content. At this stage possible inconsistencies and ambiguities related to instances in the content should be processed.

We suggest to consider the second stage of the ontology population process as work with Scott information systems. A Scott information system $T$ is a triple $(T, Con, \vdash)$, where

- $T$ is a set of tokens and $Fin(T)$ is a set of finite subsets;
- $Con$ is a consistency predicate such that $Con \subseteq Fin(T)$, $\emptyset \vdash Con$ and
  1. $Y \in Con$ and $X \subseteq Y \Rightarrow X \in Con$,
  2. $a \in T \Rightarrow a \in Con$;
- $\vdash$ is an entailment relation such that $\vdash \subseteq Con \times T$ and
  3. $X \in Con$ and $X \vdash a \Rightarrow X \cup \{a\} \in Con$,
  4. $X \in Con$ and $a \in X \Rightarrow X \vdash a$,
  5. $\forall b \in Y : X \vdash b$ and $Y \vdash c \Rightarrow X \vdash c$.

An ontology population system based on an ontology, input data, and the ontology population and the data processing rules, could be defined as a triple $(A_{op}, Con_{op}, \vdash_{op})$. Set of tokens $A_{op}$ is a set of all (underdetermined) ontology instances formed by the rules in the determination process of initial instances:

- class instances $a, \ldots$ of form $(Class_a, Atr_a)$,
- $Atr_a$ are values of attributes of class $Class_a$ containing:
  grammar information (morphological and syntactic features), and
  structural information (position in input data);
For every relation instances \( r \), . . . of form \((Relation_r, (o_1, o_2)_r, Atr_r)\):
- \((o_1, o_2)_r \) are instances of classes related by the relation \( Relation_r \),
- \( Atr_r \) are values of attributes of the relation \( Relation_r \) containing
grammar and structural information.

For every \( X \subseteq A \) let \( X = X_C \cup X_R \), where \( X_C \) is a set of class instances and \( X_R \) is a
set of relation instances.

Consistency predicate \( Con_{op} \) and entailment relation \( \vdash_{op} \) correspond to the rules of
ontology population and data processing. Let \( x, x' \in A_{op}, o \in A_C, \) and \( X \subseteq A_{op} \). The
entailment relation connects informationally associated tokens:
\[- X \vdash_{op} x, \text{iff } x \in X, \text{ or } x \notin X \land \exists y \in X_R : x \in O_y, \text{ or } x \notin X \land (Atr_x = \emptyset \lor Atr_x = \{\alpha \mid x' \in X : \alpha \subseteq Atr_r\}) \land (x \in A_R \rightarrow (O_x = \emptyset \lor \forall o \in O_x : x \vdash_{op} o)),\]
i.e. instance \( x \) is entailed from \( X \), if \( 1 \) it is in this set, or \( 2 \) it is an object of some
relation from this set, or \( 3 \) information from tokens of this set is used for evaluating
attributes or objects of \( x \), and \( x \) does not include other information.

The consistency predicate determines informationally bound sets of tokens:
\[- X \in Con_{op}, \text{if } \exists x \in A_{op} : X \vdash_{op} x, \text{i.e. a set of instance-tokens is consistent, if it}
etails (new) instance;
- \( \emptyset \in Con_{op} \).

Set \( X \) is nontrivial consistent, if \( \exists x \notin X : X \vdash_{op} x \), i.e. the set entails not only its elements and \( x \) is nontrivially entailed. An information order relation \( \prec \) is defined on
class and relation instances. Let \( a, a' \in A_C \) and \( r, r' \in A_R \):
\[- a \prec a', \text{if } a = a' \text{ everywhere except for at least one attribute, with the}
  \text{number of values of this attribute in } a \text{ being strictly less than that in } a';
- r \prec r', \text{if } r = r' \text{ everywhere except for (1) at least one object and/or}
  (2) at least one attribute, with the number of values of this attribute in } r
  \text{ being strictly less than that in } r'.\]

For token \( x \in A_{op} \): if \( x \prec x' \), then \( x' \) is information extension of \( x \), and \( x^\uparrow = \{x\} \cup \{x' : x \prec x'\} \) and \( x^\downarrow = \{x\} \cup \{x' : x' \prec x\} \) are upper and down cones of \( x \).

**Theorem 1.** Triple \( (A_{op}, Con_{op}, \vdash_{op}) \) is a Scott information system.

**Proof.** Let us show that the consistency predicate \( Con_{op} \) and the entailment relation
\( \vdash_{op} \) satisfy properties 1–5 of information systems.

1. \( Y \in Con_{op} \) and \( X \subseteq Y \Rightarrow X \in Con_{op} \). For trivial consistent sets the proposition is
obvious. Let \( Y \) is nontrivial consistent: \( Y \vdash_{op} z, z \notin Y \). Then \( \exists z' \in z^\uparrow : X \vdash_{op} z \) or
\( X \) is trivial consistent, hence \( X \in Con_{op} \).
2. \( a \in A_{op} \Rightarrow a \in Con_{op}, \{a\} \vdash_{op} a, \) hence \( \{a\} \in Con_{op} \).
3. \( X \in Con_{op} \) and \( X \vdash_{op} a \Rightarrow X \cup \{a\} \in Con_{op}. \) \( X \cup \{a\} \) is trivial consistent.
4. \( X \in Con_{op} \) and \( a \in X \Rightarrow X \vdash_{op} a. \) By the def. of the entailment relation.
5. \( \forall b \in Y : X \vdash_{op} b \) and \( Y \vdash_{op} c \Rightarrow X \vdash_{op} c. \) For trivial consistent sets the proposition is
obvious. Let sets \( X \) and \( Y \) are trivial consistent. Set \( Y \) includes information from instance-tokens of \( X \) only, hence token \( c \) does not include information more than in \( X \), hence \( X \vdash_{op} c. \) Formally: \( Y \vdash_{op} c, \) if \( c \notin Y \land (Atr_c = \emptyset \lor Atr_c = \{\alpha \mid \exists b \in Y, X' \subseteq X : \alpha \subseteq Atr_b \subseteq \cup_{x \in X} Atr_x\}) \land (c \in A_R \rightarrow (O_c = \emptyset \lor \forall o \in O_c : o \in Y \lor X \vdash_{op} Y \vdash_{op} o)), \) hence \( X \vdash_{op} c. \)

The proposition below directly follows from monotonicity of the entailment relation
and finiteness of input data.
Proposition 1. Ontology population process terminates.

Let us define a closure of an entailment relation: \( X \vdash^* x \) iff \( \exists y \in A, Y \subseteq A : X \vdash y \land Y \cup \{ y \} \vdash^* x \). An information state of set of token \( X \subseteq A \) is all tokens (all information) that can be obtained from this set by the entailment relation: \( F(X) = \{ x | \forall Y \subseteq \text{Fin}(X) \land Y \vdash^* x \} \). A projection of information state of set \( X \) to token \( a \in X \) is all tokens that can be entailed from this set using this token: \( F(X, a) = \{ x | \forall Y \subseteq \text{Fin}(X) : a \in Y \land Y \vdash^* x \} \). Let a context of token \( a \) be \( F(a) = \{ X | a \in F(X) \} \cup_{a \in X} F(X, a) \). Intuitively, a context of \( a \) is a set of token which includes all tokens necessary for the token entailment and all tokens entailed using the token. In the ontology population framework a context of an instance-token can be used for resolution of context-depending ambiguity. Suppose that there are two instance-tokens that are at the same position of input data. The ontology can be populated with only one of them. Obviously, we prefer the token that is more closely related to the other information obtained from the input data. This is the token that possesses a more powerful context which uses nontrivial entailment relations only.

Hence, the problem of context-depending ambiguity resolution is reduced to the problem of computing the cardinality of contexts for competing instance-tokens. We show that an information system of ontology population generates a special multi-agent system with typed connections. Agents of the system resolve the ambiguities by computing and comparing the context cardinalities.

3 A Multi-agent System of Ambiguity Resolution

Let a set of maximal determined instances, which is the result of the analysis of input data, be \( A^1_{\text{op}} = \{ x \in A_{\text{op}} | x^I = \{ x \} \} \). For every \( x \notin A^1_{\text{op}} \) the corresponding maximal determined instance is \( \tilde{x} \) such that \( \tilde{x} \in A^1_{\text{op}} \land x < \tilde{x} \).

Entailment relation \( \vdash_{\text{op}} \) generates information connections between maximal determined instances. Let \( X \vdash_{\text{op}} x \) and \( y \in X \land y \neq x \). Then

- attribute connections between \( \tilde{y} \) and \( \tilde{x} \) are
  - \( \tilde{y} \xrightarrow{\alpha} \tilde{x} \) iff \( \exists \alpha \in \text{Atr}_y, \alpha \in \text{Atr}_x \);
  - of tutorial type \( \tilde{y} \xrightarrow{\alpha_{\text{tut}}} \tilde{x} \) iff \( \exists x' \in X : x' < x \);
  - of parental type \( \tilde{y} \xrightarrow{\alpha_{\text{par}}} \tilde{x} \) iff \( \exists x' \in X : x' < x \);
- object connections between \( \tilde{y} \) and \( \tilde{x} \) are
  - \( \tilde{y} \xrightarrow{\text{R}} \tilde{x} \) iff \( x \in A_R \land (y \in X_C \land y \in O_x \lor y \in X_R \land \exists o_y \in O_y : o_y \in O_x) \);
  - of parental type \( \tilde{y} \xrightarrow{\text{par}} \tilde{x} \).

Information system \( (A_{\text{op}}, \text{Con}_{\text{op}}, \vdash_{\text{op}}) \) generates Multi-agent System of Ambiguity Resolution (MASAR) as a tuple \( S = (A, C, I_C, T_C) \), where

- \( A = \{ a_x | x \in A^1_{\text{op}} \} \) is a finite set of agents corresponded to maximal determined instances;
- \( C = \{ \alpha | \exists x, y \in A_{\text{op}} : \tilde{x} \xrightarrow{\alpha} \tilde{y} \} \cup \{ y | \exists x, y \in A_{\text{op}} : \tilde{x} \xrightarrow{\text{R}} \tilde{y} \} \) is a finite set of connections;
- mapping \( I_C : C \rightarrow 2^{A \times A} \) is an interpretation function of (ordered) connections between agents: \( I_C(c) = (a_x, a_y) \) iff \( \tilde{x} \xrightarrow{\alpha} \tilde{y} \).
\[ T : C \times 2^A \times A \rightarrow \{ \text{tut, par} \} \] is types of connections: \( T(c, a_x, a_y) = r \)
iff \( I_C(c) = (a_x, a_y) \rightarrow (\check{x}, \check{y}), r \in \{ \text{tut, par} \} \).

For every agent \( a \in A \) we define the following sets of agents and connections. The similar definitions are used in the graph theory, but we would like to reformulate them for the clarity. We omit symmetric definitions of ancestors \( \text{Anc}^* \) (for \( \text{Des}^* \)) and predecessors \( \text{Pred}^* \) (for \( \text{Succ}^* \)) for the brevity:

- \( C_a = \{ c \in C | \exists a' \in A : (a, a') \in I_C(c) \} \) is connections of \( a \);
- \( \text{Des}_c^a = \{ a' \in A | (a, a') \in I_C(c) \} \) is a set of descendants by \( c \) connection;
- \( \text{Des}_C^a = \bigcup_{c \in C_a} \text{Des}_c^a \) is a set of descendants;
- \( \text{Succ}_C^c = \text{Des}_C^c \cup \bigcup_{a' \in \text{Des}_C^c} \text{Succ}_a^{c_r} \) is successors by \( c \) connection;
- \( \text{Succ}_C^a = \bigcup_{c \in C_a} \text{Succ}_C^c \) is a set of successors.

The conflict resolution in MASAR is to get a conflict-free MASAR of the maximal weight. Hence \( \text{Conf} \) pair of agents from \( \text{Conf} \) set is deleted from the conflict set and the set of agents in MASAR. Hence the decrease the weight of all successors and predecessors of a conflicting agent. The first conflict pair of agents is deleted from the chain of conflicts we should know how much each conflict resolution step affects to

\[ wt(S) = \sum_{a \in A} wt(a) \]

Let conflict set \( \text{Conf} \subseteq A \times A \) be a set of unordered conflict pairs of agents corresponded to informational objects formed for the same data fragments. For every pair of agents from \( \text{Conf} \) we say that a conflict is resolved if one of the agents in the pair called the minor agent deletes itself from \( A \) (with all its connections) and involves all its descendants. Every descendant involved performs the following conflict actions:

1. if the involving connection is of the parental type then it deletes itself from \( A \) (with all its connections) and involves all its descendants;
2. if the involving connection \( c \) is of the tutorial type then it deletes all outgoing connections \( c \), and involves its descendants connected by the connections.

Note that all successors of the minor agents are involved in its conflict resolution. A conflict pair of agents is deleted from \( \text{Conf} \) after conflict resolution. Conflict actions decrease the weight of all successors and predecessors of a conflicting agent. The first conflict action reduces the conflict set and the set of agents in MASAR. Hence the system is dynamic due to conflict resolution. Change of the system weight with the fixed weight function depends on a policy of conflict actions for every agent. Problem of conflict resolution in MASAR is to get a conflict-free MASAR of the maximal weight.

We develop a multiagent algorithm that produces such system.

4 Conflict Resolution in MASAR

For constructing the conflict-free multiagent system of the maximal weight by resolving a chain of conflicts we should know how much each conflict resolution step affects to
the system weight. For every agent in conflict, it is necessary to compute its conflict weight which is the difference between the system weight before and after the agent conflict resolution. Distributed computing of this weight takes polynomial time.

Action protocols for conflict resolution used by MASAR agents forms a multi-agent system of conflict resolution MACR. The system MACR includes set of MASAR agents and an agent-master. Note, that a fully distributed version of our algorithm could be developed but it should be very ineffective. The result of agent interactions by protocols described below is the conflict-free MASAR. All agents execute their protocols in parallel until the master detects termination. The system is dynamic because MASAR agents can be deleted from the system.

The agents are connected by synchronous duplex channels. The master agent is connected with all agents, MASAR agents are connected with their ancestors and descendants. Messages are transmitted instantly via a reliable medium and stored in channels until being read.

Let $A = \{a_1, ..., a_n\}$ be a MASAR agents set, and $M$ be the master agent. Let $A_i$ be an interface protocol of agent $a_i$, and $M$ be the protocol of actions of the agent-master $M$. Then multi-agent conflict resolution algorithm MACR can be presented in pseudocode as follows:

MACR: parallel $\{A_1\} ... \{A_n\} \{M\}$

Our algorithm for constructing a conflict-free MASAR of the maximal weight is a greedy algorithm. At every step it chooses for resolution a conflict which has the maximal effect to the system weight. This effect depends on conflict actions of involved agents. Hence the following algorithms should be implemented: calculating of agents’ weights, calculating of agents’ conflict weights, the main algorithm for constructing a conflict-free set of agents of the maximal weight. Calculating the weights should be performed by MASAR agents, but constructing a conflict-free set should be conducted by the master agent.

We define an interface protocol $A_i$ for system agents, which specifies agent’s reactions for incoming messages. These messages include information which protocol $Act$ (to perform a conflict action) or $ChangeWeight$ (to change its weight) should be run and their parameters described below at the protocols’ definitions. Until an input message cause an agent to react the agent stays in a wait mode.

**Interface protocol of agent $a$.**

$A_i(a) ::$

1. set of $msg$ Input; $msg$ mess=$(start,\emptyset)$;
2. while (mess.act != Stop )
3. if( Input != null ) then {
4. mess = get (Input);
5. if( mess.act = ToAct ) then Act(mess);
6. if( mess.act = ToChange ) then ChangeWeight(mess);} 

(1) The main algorithm for conflict resolution

Let us give an informal description of protocol Master. Let $Del_A$ be set of agents removed from $A$ due to particular conflict resolution, $PartConf_a = \{b \in A|(a,b) \in Conf\}$ be a set of agents in conflict with $a$ and $Conf_A = \{a \in A|\exists b \in A(a,b) \in Conf\}$ be a set of agents in a conflict. Until the latter set becomes empty the following
steps repeat: (1) to compute of agents’ weights by launching agents to perform protocols \texttt{WeightCount} in parallel, (2) to compute of agents’ conflict weights by launching conflict agents to perform protocols \texttt{Start} in sequence, (3) to find the minor partner of the agent of maximal impact for the system weight, with the maximal difference in their conflict weights, (4) to change the weights of agents involved by this agent by launching the minor agent to perform protocol \texttt{Start}, (5) to remove the conflict of the agent and conflicts of deleted agents from the conflict set and (6) to recalculate the set of agents in conflicts. We consider the master can detect termination moments of other agents’ parallel computations at every step. The protocol of conflict weights computing and weights changing belongs to the class of wave echo algorithms [8]. Let function \texttt{max\_wConf}(\texttt{X}) (\texttt{min\_wConf}(\texttt{X})) returns the agent of the maximal (minimal) conflict weight in set of agents \texttt{X}.

\textbf{Protocol of the master agent for conflict resolution.}

\texttt{Master ::}
\begin{verbatim}
agent a, b;
1. while (ConfA ≠ ∅) {
  2. forall a ∈ A in_parallel WeightCount(a);
  3. forall a ∈ ConfA in_sequence Start(a, true);
  4. a = max\_wConf(ConfA); b = min\_wConf(PartConf_a);
  5. Start(b, false);
  6. Conf = Conf \{(a,b)};
  7. forall c ∈ DelA Conf = Conf \{(c, d) | d ∈ PartConf_c};
  8. recalculate(ConfA);
}
  9. forall a ∈ A in_parallel send (Stop) to a;
\end{verbatim}

\textbf{(2) Computing agents’ weight}

Let the set of connection of agent \texttt{a} is \texttt{C}_\texttt{a} = \{c₁, \ldots, cₙ\}. Following the definitions of the weights the agent launches in parallel calculations of the sum weight by every connection \texttt{c}_i for successors \texttt{CiDes} and for predecessors \texttt{CiAnc} (line 1) and stores calculated weights in arrays \texttt{w\_Des} and \texttt{w\_Anc} respectively. When these parallel calculations are finished, the agent computes its own weight (lines 2–3). The calculation processes have local channels \texttt{Input} for messages with integer weights of successors (predecessors). They send the weights increased by 1 to predecessors (successors) respectively. We omit the similar description of predecessors’ processes \texttt{CiAnc} for the brevity. All these agent’s weights are accessible to the agent for changing in its other protocols.

\textbf{Protocol of agent \texttt{a} for weight compute.}

\texttt{WeightCount (a) ::}
\begin{verbatim}
array [n] of int: w\_Des, w\_Anc; int w\_Des\_own, w\_Anc\_own;
1. parallel \{C1Des\} \{C1Anc\} ... \{CnDes\} \{CnAnc\}
  2. w\_Des\_own = ∑_{i ∈ 1..n} w\_Des[i]; w\_Anc\_own = ∑_{i ∈ 1..n} w\_Anc[i];
  3. wt\_A = w\_Des\_own + w\_Anc\_own + 1;
CiDes() ::
  set of int Input; NumD = |Des\_a|
1. w\_Des[i] = 0;
2. while( NumD != 0 )
\end{verbatim}
3. if ( Input != null ) then {
4. w_Des[i] = w_Des[i] + get( Input ); NumD = NumD - 1;
5. forall (b in Anc[i]) send w_Des[i]+1 to b;

(3) Computing agents' conflict weight

The next protocol is performed by the agent which starts to compute its conflict weight (wc=true) or is the minor agent (wc=false). It performs conflict action 1 launching a wave of weight changing of successors and predecessors and initiates actions of involved agents in protocol Act in line 2. Before the wave starts, agents restore its initial weights and the presence status in line 1. If there is real system change (not conflict weight computing) then all agents change their weight in line 3. Let \( wt^a \) be integer temporal weight of agent \( a \), and boolean variable \( a.Rmvd \) be true if \( a \) is removed by its conflict action 1, and false in other case.

Protocol of initial agent \( a \)

Start(\( a \), wc) ::
1. forall \( b \in A \) wt\( b=wt(b) \); \( b.Rmvd = false; \)
2. Act(1, \( \emptyset \), \( \emptyset \), wc);
3. if (not wc) forall \( b \in A \) wt\( b=wt(b) \);

An input for protocol Act is a message of the form mess = (ct, \( x \), \( c \), wc), where ct is conflict action type, \( x \) is an agent which activate this action, \( c \) is a connection with this agent, and wc is true if a conflict weight is computing. In this protocol (lines 1–2) an agent depending on the type of its conflict action (1) determines the difference of own weight, (2) forms sets of descendants and ancestors which weights are changing due to this action, and (3) specifies the amount of these changes in variable \( w \).

Then the agent sends the corresponding messages to the partners (lines 7) launching a wave of weight changing of its successors and predecessors and waits when it finishes (line 8). Further the agent depending on its conflict type launches a wave of conflict actions of its successors (lines 9, 10) and waits when it finishes (line 11). The agent has local channel Input for messages with integer conflict weights of involved successors. In line 13 the agent sums these weights. Let function \( i(c) \) returns index of connection \( c \) in set \( C_a \) for agent \( a \). \( ActId = \{1,2\} \) be a set of indexes of conflict actions, and \( C^a_1 = \{a' | \exists c : T(c,a,a') = par \} \) and \( C^a_2 = \{a' | \exists c : T(c,a,a') = tut \} \) be sets of agents immediately involved in a conflict by agent \( a \) to perform action 1 or 2.

Protocol of agent \( a \) for conflict actions.

Act( mess = (ct, \( x \), \( c \), wc) ) :: {
   int \( w \), \( wConf \), \( wConfTmp \); agent \( b \); connection \( c \);
   set of agents Desa, Anca; set of int Input; ActId \( i; \)
1. if ( ct = 1 ) then Desa = Des\( a \); Anca = Anc\( a \); \( wConf = wt_A \);
2. if ( ct = 2 ) then Desa = Des\( c \); Anca = Anc\( c \);
   \( wConf = w_Des[i(c)]+w_Anc[i(c)]; \)
3. forall b∈Desa∪Anca {
4. \( c = (a,b); \)
5. if b∈Desa then Rel = Anc; \( w = w_Anc[i(c)]; \)
6. else Rel = Des; \( w = w_Des[i(c)]; \)
7. send (ToChange, delMe, \( a \), \( c \), \( w \), Rel) to \( b; \})
8. wait (doneWt) from all b;
9. while( Input != ∅ ) wConf = wConf + get( Input );
10. if( ct = 1 ) then forall i∈ActId forall b∈C
    send (ToAct, i, a, (a,b), wc) to b;
11. if( ct = 2 ) then forall i∈ActId forall b∈C ∩ Des
    send (ToAct, i, a, c, wc) to b;
12. wait (doneAct) from all b;
13. if( wc ) then {
14. 
15. if ( x != null ) then send (doneAct, wConf) to x;}
16. else A = A \{a\}; }

An input for the next protocol is a message of the form mess = (act, x, c, w, Rel), where act specifies should the agent a remove (act=delMe) agent x from its ancestors (Rel = Anc) or descendants (Rel = Des) by connection c (lines 2,5). The agent decreases its corresponding weights by w. Decreasing of the weight affects weights of its successors and predecessors. The agent initiates the changing of these weights in line 8 and waits when it finishes (line 9).

Protocol of agent a for its weight changing.

ChangeWeight( mess = (act, x, c, w, Rel) ) :: {

    int w; agent b; set of agents Parts;
1. if( Rel = Anc ) then w_Anc[i(c)] = w_Anc[i(c)] - w;
2. if( act = delMe ) then Anc_a = Anc_a \{x\};
3. Parts = Des_a;
4. else w_Des[i(c)] = w_Des[i(c)] - w;
5. if ( act = delMe ) then Des_a = Des_a \{x\};
6. Parts = Anc_a;
7. wt_a = wt_a - w;
8. forall b∈Parts send (ToChange, decMe, a, c, w, Rel);
9. wait (doneWt) from all b∈Parts;
10. while( Input != ∅ ) wt_a = wt_a + get( Input );
11. send (doneWt, wt_a) to x; } }

Consider now the performance of the conflict resolution algorithm in a particular case of ambiguity in the following sentence:

On October 22, 2013, an official ceremony was held in the Nenets Autonomous District to mark the start of pilot oil production at the A. Titov field.

We consider Energetics as an ontology subject domain. Thesaurus of this subject area among others should contain single-word terms pilot, oil and production together with multi-word terms pilot oil and oil production. Thus the ambiguity in the example above is the following:

[[ pilot [ oil production ] ] ←→ [ [ pilot oil ] production ]]

During the multiagent algorithm initialization for the above sentence the following lexical objects L1–L5 is created with semantic attributes from the thesaurus (see Fig. 1). As a result of main stage of multiagent algorithm by the means of rule-agents implementing search of information concerning activities related to the oil production, an informational agents I1–I4 and R1–R3 corresponding to the ontological classes and relations is created.
Fig. 1. An example of conflicting agents.

Thus the main stage of the analysis in our example results in the following list of ambiguity conflicts: (L1,L2), (L2,L3), (L3,L4), (I1,I2), (I3,I4), (R1,R2). Calculated weights of agents are also depicted at Fig.1. The conflict resolution algorithm deletes agents L2, I1, and R1 at the first iteration, and L4 and I4 at the second one. The result of the algorithm is the set of information agents I2, I3, I5, R2, and R3. Thereby all remaining conflicts are resolved automatically.

5 Conclusion

In this paper, we show that instances of the ontology classes and relations that take part in the process of population form, together with the rules of data processing and ontology population, a Scott information system. This results in a simple proof of termination of the population process and justification of the resolution of context-dependent ambiguity of instances by calculating their context cardinalities. The Scott information system is a basis for our approach to context-dependent ambiguity resolution. This system generates a multi-agent system in which agents resolve the ambiguity by computing the cardinality of their contexts. The suggested algorithm of ambiguity resolution chooses the most powerful agents and removes their competitors. The choice is based on agents’ weights and their impact on the system in the process of ambiguity resolution.

In the near future we plan to give formal proofs of correctness of the algorithm proposed and to estimate its time complexity. In a development process of our multi-agent system of the semantic analysis of natural language texts for ontology population, we intend to carry out integrated testing and to rate quality of these algorithms in terms of completeness and soundness.
References


Application of Genetic Algorithms and High-Performance Computing to the Traffic Signal Setting Problem

Pawel Gora$^1$ and Przemyslaw W. Pardel$^2$

1 Faculty of Mathematics, Informatics and Mechanics
University of Warsaw
p.gora@mimuw.edu.pl

2 Interdisciplinary Centre for Computational Modelling
University of Rzeszów
ppardel@ur.edu.pl

Abstract. The paper presents results of our research on application of genetic algorithms to the problem of finding good configurations of traffic signals offsets in a road network (Traffic Signal Setting problem). We tested algorithms on a large road network - realistic map of Warsaw acquired from the OpenStreetMap project. The main research tool was the software Traffic Simulation Framework developed by the first author. To speed up experiments we employed a high-performance computing cluster at the University of Rzeszów - sessions of experiments were supervised by the second author.

Key words: traffic optimization, traffic modelling, traffic simulation, Traffic Signal Setting problem, metaheuristics, high-performance computing

1 Introduction

Large traffic congestion is an important civilizational and commercial problem, especially in urban areas. It causes travel delays, noise, stress of drivers, increase of air pollution, fuel and energy consumption, problems in organizing public transport and detours etc. It is estimated that drivers in 7 largest polish cities lose yearly approximately 3.5 billion PLN because of traffic jams [DEL]. The situation is similar in other countries, e.g., drivers in US lose yearly approximately 5.5 billion hours and 2.9 billion gallons of fuel [PDSG], it is forecasted that the cost of traffic gridlocks will be 293.1 billion dollars by 2030 (almost a 50% increase from 2013) [CEBR]. Thus, it is important to conduct a research on how to improve the traffic flow, especially in urban, densely inhabited areas.

* Experiments were carried out in the Interdisciplinary Centre for Computational Modelling. The research was partially supported by the computational grant nr 1/2015 at the University of Rzeszów and by the European Social Fund from the European Union. Authors would like to express their gratitude to Andrzej Skowron and Jan Bazan for their support during conducting research presented in the paper.
In the research traffic optimization methods based on adaptive traffic signal control are developed, with focus on the Traffic Signal Setting problem [CYP], [GOR3] in which the goal is to find the best possible configuration of offsets for a given traffic situation. For this problem a genetic algorithm (GA) is applied, in which fitness functions take into account average speed, total stopping time and total time of drive with low speed (below 20 km/h). Values of quality functions (fitness functions) of traffic signal settings are computed using microscopic and mesoscopic traffic simulation models implemented in the Traffic Simulation Framework software, developed by P.Gora ([GOR1], [GOR2]). Experiments were conducted on a realistic road network of Warsaw originating from the OpenStreetMap project [OSM]. The main novelty of the research, in comparison to previous results [GOR3], is introducing mesoscopic traffic model, calculating fitness function and modifying traffic signal settings on smaller areas, computing values of fitness functions using high-performance computing cluster.

The rest of the paper is organized as follows: Sect. 2 presents the software Traffic Simulation Framework which was used as a research tool to conduct experiments and calculate fitness function. Section 3 describes a problem of traffic optimization which we tackle, defined as the Traffic Signal Setting problem, outlines our approach to solve it and summarizes results of conducted experiments. Section 4 concludes the paper and outlines plans for the future work and possible improvements.

2 Traffic Simulation Framework

In the research, a special version of the Traffic Simulation Framework software (TSF, in short), was used for the purpose of computing values of fitness functions. TSF is an advanced tool for simulating and investigating vehicular traffic in cities, being developed at the University of Warsaw by P. Gora. It uses realistic maps acquired from the OpenStreetMap project [OSM] and is able to perform realistic, large-scale traffic simulations. The tool currently implements a microscopic traffic simulation model being extension of the Nagel-Scheckenberg model [NS]. Also, a new mesoscopic model (Sect. 2.2) has been recently developed in order to conduct even faster simulations. TSF is developed in C#.NET [NET], it possesses Graphical User Interface presenting maps and enabling editing many simulation parameters, as well as modifying maps (e.g., editing locations and configurations of traffic signals, specifying Origin-Destination (OD) matrix). It can record 5 types of output data: average speeds, congestion, instantaneous positions and velocities, locations of traffic jams, structure of the road network. The software was described in more details in the previous works [GOR1], [GOR2], which contain description of the underlying microscopic model, being extension of the Nagel-Schreckenberg model [NS]. It is important to notice that TSF has already found a few scientific applications and is still under development, e.g., recently, thanks to cooperation with Wojciech Chmiel, few new modules were implemented [CHMIEL]:

- module for reading transportation zones and OD matrices constructed using these zones, defined in the format provided by urban roads authorities in Warsaw, ZDM [ZDM];
- module for estimating OD matrix from traffic counts;
- module for calculating routes by taking into account drivers’ responses to the traffic situation.

2.1 Microscopic Model

The TSF’s microscopic simulation model is an extension of the Nagel-Schreckenberg model [NS] (Na-Sch model) based on a probabilistic cellular automaton, in which a 1-dimensional road is represented as an infinite tape divided into cells (corresponding to the area of a length 7.5 meters), time and space are discrete. At each step each cell may be empty or occupied by a single car. Cars are indistinguishable and move according to rules imitating realistic drive. The model was extensively studied and extended to more sophisticated cases, e.g. 2-dimensional road network [CS], multiple lanes [NWWS].

Na-Sch model was also extended to the case of a realistic road network, represented as a directed graph, implemented in the TSF software [GOR1], [GOR2]. The most important extensions are: many lanes, traffic signals, different profiles of drivers (influencing default maximal speed), different types of roads, speed reduction before crossroads and traffic signals, positions of cars within a single cell. In TSF routes are calculated using extension of the A* algorithm - weights of edges are estimated times of drive, based on geographical distance and default maximal speeds of drive, randomization models possible deviations from expected times.

2.2 Mesoscopic Model

Recently, a mesoscopic model was implemented in TSF, as a second traffic model (beside microscopic model). The model is not time-based, but event-based, which means that it doesn’t specify position and speed of cars at each time step, instead, it estimates time of drive (\(T_{EST}\)) on road between 2 neighbouring nodes on the path, based on the geographical distance between nodes (\(D\)) and default maximal speed on this road (\(V_{MAX}\)): \(T_{EST} = 1.2 \cdot \frac{D}{V_{MAX}}\). Then, if a car has to wait on a red signal, the total time of drive is increased by a time to the next switch from the red signal state to the green signal state.

This is a very simple mesoscopic model, constructed for the purpose of traffic optimization, to assess quality of traffic signal configurations (e.g. to minimize the total waiting time, because of a red signal). It doesn’t take into account a speed reduction caused by large traffic density (in fact, it is not straightforward to estimate traffic density on every road segment at arbitrary moment of time), queueing models and capacity models. These improvements are in plans for the further research.

3 Traffic Optimization

There are many approaches to traffic optimization and reducing impact of large traffic gridlocks, for example: adaptive traffic signal control, adaptive control of speed limits, proposing alternative routes to drivers, building new roads, introducing tolls for driving in the city center, introducing intelligent parking systems. The approach presented in the paper is focused on the first mentioned method: adaptive traffic signal control.
There already exist traffic management systems deployed in many cities, for example ZSZR [ZSZR], UTMS [UTMS], TRISTAR [TRI]. These systems implement some of the mentioned traffic optimization methods. Among adaptive traffic control mechanisms implemented in traffic management systems the most popular are: SCATS [SCA], SCOOT [SCO], OPAC [OPAC], RHODES [RHO], MARLIN [MAR], MOTION [MOTION]. These systems collect traffic data from detectors installed close to crossroads (usually: inductive loops, cameras) and adaptively change configurations of traffic signals based on measured values of congestion, lengths of queues etc. Some of these systems perform short-term traffic prediction, based on historical and actual data, to propose even better traffic control strategies. The efficiency of such traffic management systems nowadays is already quite good, but the large and even increasing traffic congestion creates demand for designing even better approaches. The rapid progress in computer science, especially in fields of machine learning, artificial intelligence, Big Data processing, and high-performance computing makes it possible to develop new traffic management solutions, potentially more effective than existing strategies.

The existing traffic management systems could be improved, for example, by collecting traffic data (positions, speeds and, potentially, also routes) using GNSS [GNSS] and V2X (V2I - vehicle-to-infrastructure or V2V - vehicle-to-vehicle) communication [V2X] and performing advanced traffic prediction based on data analysis methods (e.g., Bayesian networks [ABG]) or traffic simulations calibrated using real-world data [GOR1], [GOR2]. In response to local (predicted) gridlocks, traffic optimization methods could adapt traffic signal settings globally, or, at least, on much larger area than it is realized in most of existing traffic management systems. Traffic signal control systems on neighbouring crossroads could potentially collaborate as a self-organizing system and exchange information in order to make traffic control even better [MAR].

In the paper, we focus on one traffic optimization method, being the initial step in the process of building an advanced traffic management system of a new generation. Namely, we propose a methodology based on a genetic algorithm supported by high-performance computing cluster (to speed up computations) to find suboptimal configurations of traffic signals (relative to the so called Traffic Signal Setting problem) for a given traffic situation. The method assumes that the traffic management system possesses a complete knowledge about the traffic, i.e., initial positions, speeds and routes of all cars. This assumption is not fulfilled nowadays, but there already exist technical means to make it possible: vehicles may be equipped with devices acquiring precise location from GNSS [GNSS], e.g., Galileo system [GALILEO] supported by EGNOS [EGNOS]. Such devices may be integrated with an onboard computer or a mobile device registering acquired data, calculating parameters of drive, such as instantaneous speed, time of drive etc. In addition, the navigation system may be used to calculate the most optimal route and guide driver through this route. All these data (positions, speeds, routes etc.) could be sent to the traffic management system using dedicated short-range communication (DSRC) technology [DSRC] realizing V2X communication [V2X]. In response, the traffic management system may send to vehicles information about the current traffic state or propose optimal routes for drive. In this research information about real traffic conditions was not taken into account in calculating routes of cars, but
this approach in the routing process is also investigated by us and may be potentially used in traffic optimization.

It seems that it is already technically feasible to acquire all data required for implementing methods proposed by our research in the real-world scenario. Nowadays the realization may be still difficult and expensive, but in the future it may be just a standard.

3.1 The Traffic Signal Setting Problem

The Traffic Signal Setting problem (TSS problem) is one of problems related to adaptive traffic optimization of vehicular traffic in cities. It can be mathematically defined as follows:

- Given is a directed graph of a road network with traffic signals located in some vertices. Traffic signals are objects with attributes: duration of a red signal phase ($T_R$), duration of a green signal phase ($T_G$), offset ($T_S$) - values of these attributes may be modified.
- **Traffic Signal Setting** (TSS) is a set of values ($T_G$, $T_R$, $T_S$) for all traffic signals in a road network.
- Given are cars with starting positions in some vertices of the graph road network, static routes, rules of drive on edges.
- Given is a function $F$ which calculates quality of a traffic signal setting.
- **Goal:** Find a traffic signal setting for which the quality is optimal.

Usually it is assumed that $T_G$, $T_R$, $T_S$ may take only nonnegative integer values, interpreted as seconds (less than 1 second of difference usually doesn’t give noticeable difference in the traffic), from a finite set. In the paper we consider a case in which it is assumed that values of $T_G$ and $T_R$ are constant and equal to 58 seconds and 62 seconds, respectively (duration of a red signal phase is a bit longer to ensure safety, there is always a short period in which all signals at the crossroad are in the red phase), so the total cycle length is 120 and the only values, that may be modified, are offsets (these values may be from the set $\{0, 1, 2, 3, \ldots, 119\}$).

To ensure safety it may be assumed that traffic signals located at the same crossroad are synchronized, traffic signals at opposite entries have the same values of all attributes, while values of attributes of traffic signals at transverse entries are set complementary, so if the state of traffic signals in one direction is **green**, then the state of signals in all transverse directions should be **red**. Thus, it is sufficient to consider only 1 traffic signal per crossroad - all other should be configured in a deterministic way to ensure traffic safety. However, even in such case the space of possible solutions is large, of the size $120^n$, where $n$ is the number of crossroads with traffic signals. In case of a large city, such as Warsaw (this is our experimental case), $n$ is approximately 800 (we don’t have a full description of the real road network, since maps used for experiments contain information about only 291 crossroads with traffic signals). It is impossible to evaluate all configurations.

What may be the quality function $F$? It depends on what is the goal of traffic optimization. Some possible functions are defined by:
- total waiting time - times spent with a speed \(0\frac{km}{h}\), summed up over all cars;
- total time spent with low speed, below \(20\frac{km}{h}\);
- average speed of all cars;
- delay - the total wasted time, in comparison to the free flow case;
- total emission of fumes;
- total fuel consumption;
- total length of queues at crossroads;
- total number of stops;

Values of these functions could be calculated by running computer simulations of the model evolution. However, realistic traffic models are often nondeterministic, so computed values in such models may come, in fact, from random variable distributions, which are usually not known explicitly. In such case, the natural choice for values of quality functions are mean values, estimated by averaging outcomes of few computer simulations (Monte Carlo method).

Some of mentioned quality functions are correlated, i.e., if values of one function increase, it usually means that values of another function also increase or decrease, but it is not straightforward to give a precise correlation. Since some quality functions may require more complicated models (fumes emission, fuel consumption) than other quality functions (stopping times, number of stops), it is better to focus on simpler functions, which might give better approximations of real-world values. The presented research has been focusing so far on 3 quality functions: total waiting time, total time spent with speed lower than \(20\frac{km}{h}\), average speed of all cars - denoted as \(T_0, T_{20}, AvgSpeed\), respectively.

A natural question is whether investigated quality functions can be calculated without running computer simulations? It turns out that traffic simulation models are usually too complicated to calculate quality of a given traffic signal setting explicitly, without running a computer simulation, i.e., it is impossible to give a compact formula for calculating the quality in a constant time, or to design algorithm which will calculate the quality in a shorter time than by running traffic simulation. This property is called \textit{computational irreducibility} and was introduced by Steven Wolfram [WOL]. The property is typical for most nontrivial cellular automata. In fact, even for very simple models, such as Nagel-Schreckenberg model [NS], it is impossible to obtain all properties without running computer simulation (with the exception of \(V_{MAX} = 1\) [SHA]). Thus, running simulations is currently the only known option to compute values of mentioned quality functions in realistic traffic models.

On the other hand, quality functions could be approximated by functions which values could be calculated in a simpler way than by running time-consuming computer simulations. If such approximations are \textit{monotonic}, in a sense that traffic signal settings with lower values of quality functions have also lower values of approximations, then computing approximations is sufficient to find optimal traffic signal settings. However, it is very difficult to find monotonic approximations. But even if approximation is not monotonic, it may be still useful. Since the space of possible solutions of the Traffic Signal Setting problem is large, it may be difficult to find the most optimum setting even if a quality function could be computed very fast. However, quality functions are just approximations of corresponding values in the real-world traffic, the optimal value of a
quality function and optimal TSS in the given traffic model may not be optimal in case of a real-world traffic. In addition, in case of a real-world traffic and nondeterministic models it may not be proved that the given setting is optimal, it may be even difficult to define what optimality actually means. Instead, it may be sufficient to find settings which are suboptimal, for which values of quality functions may be a bit worse than for (unknown) optimal configuration, but should be good enough in the model, can be accepted in reality and could be found relatively easy. Thus, it is sufficient to have approximation functions of quality functions, which will be computed fast and will guarantee that traffic signal settings with good (suboptimal) values of the approximation function will have also good (suboptimal) values of the corresponding quality function in the model and in the real-world scenario. In case of quality functions computed in the microscopic traffic simulation model, a good approximation may be the relevant function in the mesoscopic model. This is a reason why the mesoscopic traffic model (Sect. 2.2) approximating microscopic model may be considered as potentially good model for running traffic simulations and computing quality functions in case of our research. This is one of important innovations introduced in the research in comparison to the previous results [GOR3].

### 3.2 A Genetic Algorithm for the TSS Problem

For the Traffic Signal Setting problem we propose a genetic algorithm being extension of the algorithm introduced by [CYP]. The extensions are related to:

- More realistic microscopic traffic model (rules of drive based on the Na-Sch model, crossroads with more than 2 traffic signals, realistic Origin-Destination matrices, routes are known at the moment of start of drive, different profiles of drivers, different types of roads) and mesoscopic model
- Possibility to encode not only offsets, but also duration of a red signal phase and green signal phase;
- Different selection procedure - square root selection (however, other methods are also allowed)
- Many possible fitness functions, calculating fitness function on a selected area (potentially smaller than the whole road network)
- Modifying traffic signal settings on a selected area (potentially smaller than the whole road network)

### 3.3 Genetic Algorithm Specification

In the proposed algorithm TSS is represented as a genotype - vector of genes, in which each gene (vector position) corresponds to a single traffic signalization (representant of a single crossroad with traffic signals, other traffic signals on that crossroad should be deterministically synchronized to ensure traffic safety (see Sect. 3.1)). The value of a single gene represents traffic signal offset - time (in seconds) from the reference point in time to the next phase switch from the red signal state to the green signal state. This value should be from the set \( \{0, 1, 2, \ldots, T_{\text{CYCLE}} - 1\} \), where \( T_{\text{CYCLE}} \) is a duration
of a traffic light cycle equal to \( T_G + T_R = 120 \) (we assume \( T_G = 58, T_R = 62 \), see Sect. 3.1).

The length of a genotype depends on the number of crossroads considered in the experiment. We investigated 2 areas for modifying traffic signal settings and calculating fitness functions:

- A - The whole road network (291 crossroads with traffic signals)
- B - Smaller area corresponding to the Stara Ochota district in Warsaw (15 crossroads with traffic signals)

In experiments the initial population consisted of 100 or 400 genotypes, generated randomly, i.e., every gene at each position had a value being random number from the set \( S_1 = \{0, 1, 2, \ldots, 119\} \). However, in some experiments sets of possible values were sparse: \( S_5 = \{0, 5, 10, 15, \ldots, 115\} \), \( S_{10} = \{0, 10, 20, \ldots, 110\} \).

3 different fitness functions were considered: \( Time_0 \) (total stopping time - time with speed \( 0 \) km/h), \( Time_{20} \) (total time of drive with speed below \( 20 \) km/h), \( AvgSpeed \) (average speed of all cars). Values of fitness functions were calculated by running traffic simulations in the Traffic Simulation Framework. 2 different simulation models were applied: the original microscopic model and the new mesoscopic model (Sect. 2.2).

As a selection operator, the Square-root operator was chosen: from a given population with \( n \) genotypes only \( \sqrt{n} \) were selected to take part in the reproduction. The reason for that was to get rid of poor traffic signal settings as soon as possible and enhance TSSs which are better and may lead to better solutions.

As a crossover operator the Uniform Crossover operator was chosen: \( \sqrt{n} \) genotypes were crossed with each other to obtain a new set of \( n \) genotypes, and for each pair of crossing genotypes and each vector position a gene value was selected from a randomly chosen genotype.

As a mutation operator the Uniform Mutation operator was chosen: for each gene with a given probability (3 values were investigated: \( \{\frac{1}{100}, \frac{1}{20}, \frac{1}{5}\} \) the gene value was set to a value selected randomly (with a uniform distribution) from the set of possible values.

3.4 Experiments

We prepared and performed many series of experiments. In the paper, we present only experiments being milestones in the research (there were many more auxiliary experiments conducted, which led to conclusions taken into account in designing next experiments) and focus on the last experiment, which gave the best outcome so far.

1. TSS modified on area A, microscopic stochastic simulation model, fitness functions computed on area A (without HPC cluster) 5 times and averaged
2. TSS modified on area B, microscopic stochastic simulation model, fitness functions computed on area A once (without HPC cluster)
3. TSS modified on area A, microscopic stochastic simulation model, fitness functions computed on area B once (on HPC cluster)
4. TSS modified on area B, microscopic stochastic simulation model, fitness functions computed on area B once (on HPC cluster)
5. TSS modified on area B, microscopic stochastic simulation model, fitness functions computed on area B (on HPC cluster), 5 times and averaged
6. TSS modified on areas A and B, mesoscopic deterministic simulation model, fitness functions computed on areas A and B (on HPC cluster)

In each experiment, we calculated a relative difference between values of the best genotype in the first iteration and the best genotype found in all iterations. First 2 sessions of experiments were conducted on a few machines in the computer laboratory at the University of Warsaw. Because of large computational complexity of traffic simulations and genetic algorithms, we decided to use a high-performance computing cluster at the University of Rzeszów (Sect. 3.5).

Table 1. Simulation parameters used in the 6th session of experiments

<table>
<thead>
<tr>
<th>Name of the parameter</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration of traffic simulation</td>
<td>600 seconds</td>
</tr>
<tr>
<td>Initial number of cars in the simulation</td>
<td>10000, 30000, 50000, 70000</td>
</tr>
<tr>
<td>Number of cars which start drive at each step</td>
<td>10</td>
</tr>
<tr>
<td>Interval between possible values of offsets</td>
<td>1 (Set $S_1$), 10 (Set $S_{10}$)</td>
</tr>
<tr>
<td>Optimized value</td>
<td>Time0</td>
</tr>
<tr>
<td>Number of iterations</td>
<td>50</td>
</tr>
<tr>
<td>Population size</td>
<td>100, 400</td>
</tr>
<tr>
<td>Area on which TSSs are modified</td>
<td>A (the whole road network), B (smaller region, &quot;Stara Ochota&quot; district)</td>
</tr>
<tr>
<td>Area on which fitness function is computed</td>
<td>A (the whole road network), B (smaller region, &quot;Stara Ochota&quot; district)</td>
</tr>
<tr>
<td>Probability of mutating a single gene</td>
<td>0.01, 0.05, 0.2</td>
</tr>
</tbody>
</table>

The first session of experiments and its results are described in the paper [GOR3]. One of conclusions after the first experiment was that GA brings promising results - 3.11% improvement of the Time0 fitness function and 1.82% improvement of the Time20 fitness function, after only 9 iterations of the algorithm, but computations should be speed up in order to compute more iterations. This was one of goals of 4 next sessions of experiments in which we investigated different configurations of areas for calculating fitness functions and modifying traffic signal settings. However, improvements were not as good as expected, giving still only up to 5 − 6% of improvement (usually much less), despite of running much more iterations of algorithms (up to 50 populations). The reason of not satisfactory performance of GA in such models is still investigated (one of potential reasons may be influence of nondeterminism of the model on values of fitness functions). In the meanwhile, we decided to implement a new traffic model, mesoscopic model described in Sect. 2.2.

In the 6-th session we ran in parallel in the cluster 192 experiments with fitness functions computed using a mesoscopic model from Sect. 2.2 and results were significantly
better than before. The Table 1 presents values of parameters used in our experiments (in this session we used only a subset of values of parameters used in all session, e.g., the only calculated fitness function is $\text{Time}_0$, because in the mesoscopic model $\text{AvgSpeed}$ should lead to the same TSSs as $\text{Time}_0$, while $\text{Time}_{20}$ cannot be computed). We drew the following conclusions:

1. Calculating values of the fitness function $\text{Time}_0$ in the mesoscopic model takes few seconds which is about 100 times faster than in the microscopic model. It can be even improved by introducing concurrency in computing waiting times of cars (the speed up may depend on number of cars, traffic model, road network topology, architecture of the cluster, number of cores etc.) and by parallelizing computation of a single fitness function (the speed up may depend on number of genotypes in a population, architecture of the cluster, number of cores etc.). We estimate that thanks to such parallelism execution time of the presented genetic algorithm may be reduced from few hours to few minutes on the experimental cluster (Sect. 3.5) and to maximum few seconds on larger HPC clusters, making possible realtime applications.

2. If signals are modified on smaller area (B) and fitness function is calculated on the whole road network (area A), the improvement is small, in the range $0, 4657\% - 0, 7182\%$. The main reason is that modifying traffic signals on small area has small impact on global traffic parameters.

3. If signals are modified on smaller area (B) and fitness function is computed on smaller area (B), the improvement is significant, in the range $6, 8567\% - 15, 5888\%$, depending on values of other parameters.

4. If signals are modified on the whole road network (area A) and fitness function is computed on the whole road network (A), the improvement is also significant, in the range $5, 7101\% - 18, 1204\%$, depending on parameters.

5. If signals are modified on the whole road network (A) and fitness function is computed on area B, the improvement is in the range $26, 6716\% - 51, 4472\%$ (much better than in case of reconfiguring traffic signals only on the area B). However, it may cause larger delays on the rest of the road network, thus this method alone cannot be applied, but should be further investigated.

### 3.5 Computational Cluster

Experiments on a high-performance computing cluster “PEGAZ” (produced by Hewlett-Packard) were carried out in the Interdisciplinary Centre for Computer Modelling at the University of Rzeszów from January to June 2015 and were supervised by the second author (Dr. P. Pardel). The cluster consists of 40 computational nodes with 2 processors (INTEL Xeon E5-2620, 6 cores, 2.0 GHz, 15MB cache) per node. The computational power is 7.5 TeraFLOPS, 1TB RAM (1032GB, 258x 4GB DDR3 1333MHz).

### 4 Conclusions and Future Work

In the paper, new results of applications of genetic algorithms for the Traffic Signal Setting problem were presented. In comparison to previous results [GOR3] the major
improvement in computation time came from applying a high-performance computing cluster (Sect. 3.5) and introducing mesoscopic traffic simulation model, instead of microscopic model. The latter change brought also much better results of GA: more than 10% reduction of the total waiting time on the whole road network of Warsaw and on a smaller district.

As for the future research, the mesoscopic model implemented in the TSF software should be extended in order to take into account traffic congestion, queueing model and capacity model. For ensuring realism of results, a detailed calibration and validation of the mesoscopic model should be performed, by comparison to the microscopic model and using real-world traffic data, which have been recently acquired thanks to collaboration with ZDM [ZDM].

References


Lattice Theory for Rough Sets - An Experiment in Mizar

Adam Grabowski
Institute of Informatics, University of Bialystok
Ciolkowskiego 1M, 15-245 Bialystok, Poland
adam@math.uwb.edu.pl

Abstract. Rough sets is a well-known approach to incomplete or imprecise data. In the paper we briefly report how this framework was successfully encoded with the help of one of the leading computer proof assistants in the world. Thanks to their abstract and flexible character based essentially on binary relations, lattices as a basic viewpoint appeared a very feasible one in the case of rough sets. We focus on lattice-theoretical aspects of rough sets to enable the application of external theorem provers like EQP or Prover9 as well as to translate them into TPTP format widely recognized in the world of automated proof search. We wanted to have a clearly written, possibly formal, although informal as a rule, paper authored by a specialist from the discipline another than lattice theory. It appeared that Lattice theory for rough sets by Jouni Järvinen [11] (LTRS) was quite a reasonable choice to be a testbed for the current formalization both of lattices and of rough sets in Mizar.

1 Introduction

Through the years ordinary set theory appeared not to be feasible enough for modelling incomplete or imprecise information. Even if basically built on top of Zermelo-Fraenkel widely accepted by most mathematicians, fuzzy sets by Zadeh [22] proposed new view for membership functions, where degree of membership taken from the unit interval was considered rather than classical discrete bipolarity. Pawlak’s alternative approach [16], although essentially of the same origin, was different – its probabilistic features were underlined. Also the focus was put rather on collective properties of clusters of objects than those of individuals as the latter can be hardly accessible.

Formalization is doing mathematics in a language formal enough to be understandable by computers [19] (of course, doing mathematics means also the act of proving theorems and correctness of definitions according to classical logic and Zermelo-Fraenkel set theory). This activity, obviously without the use of computers is dated back to Peano and Bourbaki as every mathematician uses more or less formal language [1]; but computer certification of mathematics can be useful for many reasons – machines open new possibilities of information analysis and exchange, they can help to discover new proofs or to shed some light on approaches from various perspectives; with the help of such automated proof assistants one can observe deeper connections between various areas of mathematics. For example, lattice theory delivers interesting and powerful algebraic model – useful in quantum theory, logic, linear algebra, and topology, to list only most popular ones. Hence it is not very surprising that also rough [11] and fuzzy set theories
[22] can be modelled in this way. Studying connections between theories can be also
benefitting to lattice theory itself – see [3] for the use of upper and lower rough ideals
(filters) in a lattice.

The early works devoted to the formalization of mathematics with the help of auto-
mated proof assistants were not really connected with lattice theory. Checking Landau’s
Grundlagen... was an experiment of translating arithmetic into AUTOMATH, one of the
primary computerized proof-checkers. But first real widespread (at least for ordinary
people) use of automated theorem provers was the solution of the Robbins problem (al-
ternative axiomatization of Boolean algebras) with the help of EQP/Otter program by
William McCune, which expressed the essence of computational power of computers
in the area of equational proof search within lattice theory.

For years, there were essentially two types of activities in the computer certification
of mathematics: either formal exploration of a single important theorem, in style of
Kepler conjecture, or the computer encoding of a book or paper – all these aimed at
building large formal repository of interconnected facts formally proven. In case of
more compact research articles, the number of such efforts is quite big (even in the
Mizar Mathematical Library (MML) there are at least twenty such encoded papers), but
if the number of pages is relatively large, such projects are still rare.

The author was personally involved in the formalization projects of Compendium of
Continuous Lattices (CCL) by Gierz et al. (approximately 70% done); he also provided
more or less complete translations of papers into Mizar (Isomichi’s about classification
of subsets of topological spaces, Zhu’s [23] on the approximation spaces based on ar-
bitrary binary relations, and Rao’s on the generalized almost distributive lattices). The
author of the present paper was interested in lattice theory, focusing on formalization
issues of these structures with the help of automated proof-assistants of various kinds,
including computerized theorem provers, dedicated software for calculations and vi-
sualization, and also computer algebra systems [4]. The idea of this work arose after
having a look for Jouni Järvinen’s paper Lattice theory for rough sets [11] (which will
be called LTRS in short), hence the title of the current paper is not accidental. Our short
term goal is to provide annotated version of LTRS, where all items (definitions, propo-
sitions, and illustrative examples) will be mapped with their formal counterpart written
in Mizar.

The structure of the paper is as follows: in the next section we introduce basic
notions of lattice theory, also in mechanized setting, and discuss how it can be extended.
Section 4 is devoted to concrete implementation of rough sets with Mizar system. The
next section summarizes the pros and cons of our implementation and experiments with
the Mizar Mathematical Library. We describe also the current state of injecting LTRS
into MML. In the last section we draw some concluding remarks and plans for future.

2 Fundamentals of Lattice Theory

Lattices [10] are structures of the form

\( \langle L, \sqcap, \sqcup \rangle \),
where $L$ is a set (sometimes assumed to be non-empty), both binary operations $\sqcup$ and $\sqcap$ are commutative, associative, and satisfy the absorption laws. There is also an alternative definition of lattices as $(L, \leq)$, where $\leq$ is the partial ordering on $L$ with the existence of suprema and infima for arbitrary pairs of elements of $L$. Essentially then, one can see lattices as $(L, \sqcup, \sqcap, \leq)$, where both parts are defined by one another.

It is worth noticing that lattices, especially those of them which have equational characterizations, can be automatically explored. Famous question on another axiomatization of Boolean algebras, known as the Robbins problem, was solved with EQP/OTTER system in 1996 after sixty years of unsuccessful human research. The first author provided also some proof developments in this problem, but with the use of another computer proof-assistant, namely the Mizar system [9]. It was created in the early seventies of the previous century in order to assist mathematicians in their work. Now the system consist of three main parts: the language in which all the mathematics can be expressed, close to the vernacular used by human mathematician, which at the same time can be automatically verified, the software which verifies the correctness of formalized knowledge in the classical logical framework, and last but not least, the huge collection of certified mathematical knowledge – the Mizar Mathematical Library (MML).

In Mizar formalism [9], lattices are structures of the form

\[
\text{definition}
\begin{align*}
\text{struct} & (\setminus\mathrm{SemiLattRelStr}, \setminus\mathrm{SemiLattRelStr}, \mathrm{LattStr}) \\
\text{LattRelStr} & \\
& (\# \text{ carrier} \rightarrow \text{set}, \\
& \text{L_join, L_meet} \rightarrow (\text{BinOp of the carrier}), \\
& \text{InternalRel} \rightarrow (\text{Relation of the carrier #}));
\end{align*}
\text{end;}
\]

introduced by the first author to benefit from using binary operations and orderings at the same time. However, defining a structure we give only information about the signature – arities of operations and their results; specific axioms are needed additionally. Even if all of them can be freely used in their predicative variant, definitional expansions proved their usefulness.

\[
\text{definition}
\begin{align*}
\text{let} & \text{ L be non empty LattStr;} \\
\text{attr} & \text{ L is meet-absorbing means} :: \text{LATTICES:}\text{def 8} \\
& \text{for a,b being Element of L holds} (a \setminus \sqcap b) \sqcup b = b;
\end{align*}
\text{end;}
\]

The above is faithful translation of
\[
\forall a,b \in L \ (a \cap b) \sqcup b = b.
\]

Continuing with all other axioms, we finally obtain lattices as corresponding structures with the collection of attributes under a common name Lattice-like.

\[
\text{definition}
\begin{align*}
\text{mode} & \text{ Lattice is Lattice-like non empty LattStr}; \\
\text{end;}
\end{align*}
\]
The alternative approach to lattices through the properties of binary relations is a little bit different, so the underlying structure is just the set with \text{InternalRel}, namely \text{RelStr}.

\textbf{definition}
mode Poset is reflexive transitive antisymmetric RelStr;\end;

Boolean algebras, distributive lattices, and lattices with various operators of negation are useful both in logic and in mathematics as a whole, it is not very surprising that also rough set theory adopted some of the specific axiom sets – with Stone and Nelson algebras as most prominent examples.

The type \text{LATTICE} is, unlike the alternative approach where \text{Lattice} Mizar mode was taken into account, a poset with binary suprema and infima. Both approaches are in fact complementary, there is a formal correspondence between them shown, and even a common structure on which two of them are proved to be exactly the same. Why can ask the question why to have both approaches available in the repository of computer verified mathematical knowledge available at the same time? The simplest reason is that the ways of their generalizations vary; in case of posets we could use relational structures based on the very general properties of binary relations (even not necessarily more general, but just different – including equivalence relations or tolerances); equationally defined lattices are good starting point to consider e.g. semilattices or lattices with various additional operators – here also equational provers can show their deductive power. In this setting such important theorems as Stone’s representation theorem for Boolean algebras was originally formulated.

The operations of supremum and infimum are "\lor" and "\land", respectively – in both approaches. The natural ordering is \( \leq \) in posets and \( [=\) in lattices equationally defined. Note that \( c=\) is set-theoretical inclusion. The viewpoint of posets was extensively studied in Mizar during the big formalization project – translating into Mizar already mentioned CCL.

3 Extending Lattice Signature

From the informal point of view, the difference between the ordering given for \( \langle L, \sqcup, \sqcap \rangle \) as

\[ x \leq y \iff x \sqcap y = y \]

and posets defined as \( \langle L, \leq \rangle \) where the existence of binary suprema and infima is axiomatically guaranteed is not very big. Here is the place for Leibniz’s Law (known also under the name of equality of indiscernibles or – slightly misleading – isomorphic copies). If it comes for real computerized proof-assistant, the problem arises. First of all, the choice of appropriate signature for lattices is important. On the one hand, we want to gather benefits from the use of sophisticated equational theorem provers, and hence equational characterization is strongly desirable. We can do so choosing lattice structure with two binary operations of supremum and infimum.

On the other hand however, one can use the properties of binary relations weaker than the partial ordering – either preorders or even, if we go further in the stream of
reverse mathematics, arbitrary relations with certain properties. We can observe similar
ideas in the theory of rough sets, with the example of Zhu’s papers [23], or topological
spaces with various separation axioms.

All basic formalized definitions and theorems can be tracked under the address

4 Rough Sets and Approximation Spaces

Main disadvantage behind formalization of rough set theory [16] was that we had to
choose among two basic approaches to rough sets: either as classes of equivalence re-
lations (with further generalizations into tolerances or even arbitrary binary relations)
or as pairs consisting of the lower and the upper approximation (see [7] for detailed
description of approximation operators in Mizar).

\[\text{definition let } X \text{ be Tolerance Space, A be Subset of } X; \]
\[\text{func } RS A \rightarrow \text{ RoughSet of } X \text{ equals } \quad \text{:: INTERVA1:14} \]
\[\text{[LAp A, UAp A];} \]
\[\text{end;} \]

\[\text{The structure is defined as follows (}_\lor_\text{ and } _\land_ \text{ are taken componentwise):}\]

\[\text{definition let } X \text{ be Tolerance Space; } \]
\[\text{func RSLattice } X \rightarrow \text{ strict LattStr means } \quad \text{:: INTERVA1:23} \]
\[\text{the carrier of it } = \text{ RoughSets } X \& \]
\[\text{for } A, B \text{ being Element of RoughSets } X, \]
\[\text{A1, B1 being RoughSet of } X \text{ st } A = A1 \& B = B1 \text{ holds} \]
\[\text{(the L_join of it)}.(A,B) = A1 _\lor_ B1 \& \]
\[\text{(the L_meet of it)}.(A,B) = A1 _\land_ B1; \]
\[\text{end;} \]

\[\text{We have proved formally that these structures are Lattice-like, distributive, and complete (arbitrary suprema and infima exist, not only binary ones). The properties are expressed in the form allowing automatic treatment of such structures.}\]

\[\text{registration let } X \text{ be Tolerance Space; } \]
\[\text{cluster RSLattice } X \rightarrow \text{ bounded complete; } \]
\[\text{end;} \]

\[\text{Taking into account that } [= \text{ is the ordering generated by the lattice operations, we can prove that it is just determined by the set-theoretical inclusion of underlying approximations.}\]

\[\text{theorem } :: \text{ INTERVA1:71 } \]
\[\text{for } X \text{ being Tolerance Space, } A, B \text{ being Element of RSLattice } X, \]
\[\text{A1, B1 being RoughSet of } X \text{ st } A = A1 \& B = B1 \text{ holds} \]
\[\text{A } [= \text{ B iff} \]
\[\text{LAp A1 } \subseteq \text{ LAp B1 } \& \text{ UAp A1 } \subseteq \text{ UAp B1;} \]
Detailed survey of the lattice-theoretical approach to rough sets is contained e.g. in Järvinen [11] paper, which enumerated some basic classes of lattices useful within rough set theory – e.g. Stone, de Morgan, Boolean lattices, and distributive lattices, to mention just the more general ones. All listed structures are well represented in the MML.

Flexibility of the Mizar language allows for defining new operators on already existing lattices without the requirement of repetitions of old structures, e.g. Nelson algebras are based on earlier defined de Morgan lattices (which are also of the more general interest), also defining various negation operators is possible in this framework. Furthermore, topological content widely represented in the Mizar Mathematical Library helped us to have illustrative projections into other areas of mathematics [5]. Here good example was (automatically discovered by us) linking with Isomichi classification of subsets of a topological space into three classes of subsets [6].

Recently we made some results in fuzzy numbers in formalized setting, and even if primarily we did not plan to include them here, LTRS forced us to do so. Namely, Example 141, p. 476 in LTRS is just the construction of $L$-fuzzy sets, so it seems we have to cover it also. Exactly just like in the case of rough sets (where we tried to stick to equivalence relations instead of the more general case), first formalization in Mizar had to be tuned to better reflect Zadeh’s mathematical idea [22]. The basic object $\text{RealPoset}$ is a poset (a set equipped with the partial ordering, i.e. reflexive, transitive, and antisymmetric relation) – unit interval with the natural ordering defined of real numbers. We use Heyting algebras as they are naturally connected with $L$-fuzzy sets. We can define the product of the copies of the (naturally ordered) unit intervals (by the way, this was defined in Mizar article $\text{YELLOW}_1$ by the first author during the formalization of CCL).

**definition** let $A$ be non empty set;
func $\text{FuzzyLattice } A$ -> Heyting complete LATTICE equals :: $\text{LFUZZY}_0$:def 4
(RealPoset [. 0,1 .]) |^ A;
end;

The above can be somewhat cryptic, however the next theorem explains how elements of the considered structure look like – these are just functions from $A$ into the unit interval.

**theorem** :: $\text{LFUZZY}_0$:14
for $A$ being non empty set holds
the carrier of $\text{FuzzyLattice } A$ = Funcs (A, [. 0, 1 .]);

Although the lattice operations are not stated explicitly here (which could be a kind of advantage here), the ordering determines it uniquely via the natural ordering of real functions. The underlying structure is a complete Heyting algebra. Of course, the type of this Mizar functor is not only declarative – it had to be proved.

**theorem** :: $\text{LFUZZY}_0$:19
for $C$ being non empty set,
$s,t$ being Element of $\text{FuzzyLattice } C$ holds
$s "\lor" t = \text{max}(\emptyset s, \emptyset t);$
The functor $\Theta$ returns for an element of the lattice of fuzzy sets the corresponding membership function (i.e. a fuzzy set). The so-called type cast is needed to properly recognize which operation $\max$ should be used – in our case it is the operation defined for arbitrary functions. Of course, in the above definition, $A$ is an arbitrarily chosen non-empty set; it can be replaced by the carrier of a lattice, and the definition of $\text{FuzzyLattice } A$ can be tuned accordingly.

5 Where Are We Now

At the beginning of expressing things fully formally, usually it is hard to estimate the real amount of needed work. The unreasonable delays are often caused by authors’ omissions in proof steps, where inferences suggested to be trivial appear to be hard or even false (we met some such situations during CCL project), references for papers yet classical in the literature of the subject (but absent in the formal repository, which has to be overworked), and some isomorphisms in the spirit of category theory (let us assume continuous mappings to be arrows in proper category) or logic (all we just considered is trivial in the framework of proper modal logic).

As basic stages of our work we could enumerate:

- Choosing appropriate model for formalization – in CCL we decided to have two series, YELLOW (bridging the gap between the current state of MML and needed knowledge) and WAYBEL formalizing one-by-one items from the book. It appears however, that due to highly self-containing nature of LTRS a kind of YELLOW series practically vanishes and we will give rather a set of monographical articles (proofs from MML will be only linked, not rewritten).
- Providing a platform for the work, with the aim of incorporating the work into the Mizar Mathematical Library (we want to make it available to other users making it distributed with every distribution of the Mizar system).
- Wikipedia proved that in case of wide projects a model of Wiki for collective work wins with svn or cvs (concurrent version systems) with closed architecture (although a bit of centralized control to ensure the appropriateness of changes is definitely needed).
- Back-revisions of the MML, as the Mizar repository can also benefit (generalizations, removing repetitions, making unifications). Here we mention also the possibility of proof reorganization: lemmas can be extracted, proof structure can be flattened, if possible, etc.
- Translation of the Mizar source code into \LaTeX via available tools and comparing the differences with LTRS. During the process of printing the journal *Formalized Mathematics* which is a translation between vocabulary file and \LaTeX format given in XML; formats are once fixed, but it is no problem at all to get instead of UAp(A) the set $A$ with the upper triangle (the notation used in LTRS), etc. Of course, alternatively one can define its own synonym, which does not bring any new semantical information. We can mention also hyperlinked version of articles, hence anyone can dig down to the fundamentals.
Statistical research on results, once we get complete formal translation. Hence besides explicit references, all connections can be listed – facts from mathematical folklore, informal omissions, detailed proof steps, and proof similarities.

As of now, we do not provide percentage of the real work done (we plan eventually to give a one-to-one correspondence between LTRS and MML, and as of now there is no fully annotated version accepted into the MML), but thorough insight into both sources results in the following numbers (see Table 1).

### Table 1. LTRS chapters represented in MML

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Items</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.</td>
<td>Basic notions and notation</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3.</td>
<td>Orders and lattices</td>
<td>21</td>
<td>18</td>
</tr>
<tr>
<td>4.</td>
<td>Distributive, Boolean, and Stone lattices</td>
<td>20</td>
<td>14</td>
</tr>
<tr>
<td>5.</td>
<td>Closure systems and topologies</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>6.</td>
<td>Fixpoints and closure operators on ordered sets</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>7.</td>
<td>Galois connections and their fixpoints</td>
<td>22</td>
<td>16</td>
</tr>
<tr>
<td>8.</td>
<td>Information systems</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>9.</td>
<td>Rough set approximations</td>
<td>21</td>
<td>9</td>
</tr>
<tr>
<td>10.</td>
<td>Lattices of rough sets</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>166</td>
<td>112</td>
</tr>
</tbody>
</table>

There are three items we should briefly explain: MML lacks the part devoted to Stone lattices (Chapter 4), although we have some 1500 lines of Mizar code covering most of the missing content. Although the chapter with Galois connections is covered in 70%, it is not very much used in the Mizar repository, hence low numbers in Chapter 9 and 10. Also information systems already formalized in Mizar need to be adjusted in order to provide smooth correspondence with reducts and rough sets. In total, nearly 70% of items are covered in the current MML. In the next table, we present most representative Mizar articles (MML identifiers, i.e. filenames) for every chapter.

The quotation from [11] “Prerequisites are minimal and the work is self-contained” was very important for us – as after the finish we will calculate the so-called de Bruijn factor which measures the ratio between the amount of formal text and its informal counterpart. It is often claimed that in the case of ordinary mathematical submission its de Bruijn factor is approximately four – but it would not be the case of LTRS, as it has many illustrative examples.

The enumeration is continuous and examples are present in this sequence (in CCL, some examples were not formalized by us as either they were needed to solve the exercises, or they had purely illustrative character). Essentially then, the numbers are not fully reflecting the real state of formalization. Similar situation was during formalization of CCL in Mizar, where $C^*$-algebras were used as one of the examples (and we did...
Table 2. Representation of LTRS chapters by Mizar articles

<table>
<thead>
<tr>
<th>Chapter</th>
<th>MML Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>N/A</td>
</tr>
<tr>
<td>2.</td>
<td>XBOOLE_1, RELAT_1</td>
</tr>
<tr>
<td>3.</td>
<td>ORDERS_2, LATTICE3</td>
</tr>
<tr>
<td>4.</td>
<td>LATTICES, LATTICE2</td>
</tr>
<tr>
<td>5.</td>
<td>PRE_TOPOC, TOPS_2</td>
</tr>
<tr>
<td>6.</td>
<td>KNASTER, YELLOW_2</td>
</tr>
<tr>
<td>7.</td>
<td>WAYBEL_1, WAYBEL34</td>
</tr>
<tr>
<td>8.</td>
<td>ARMSTRNG</td>
</tr>
<tr>
<td>9.</td>
<td>ROUGHS_1, ROUGHS_2</td>
</tr>
<tr>
<td>10.</td>
<td>INTERVA1, ROUGHS_4</td>
</tr>
</tbody>
</table>

not touch it at all as it would require massive work within completely different area of mathematics we dealt with).

Many theorems and definitions are inline, i.e. they are not formulated as separate items. Extremal example here is the list of boolean properties of sets (some 17 properties collected in Proposition 1), which of course in the MML is of the form of 17 separate theorems. We could choose either between the corollary which is a conjunct of all these or mapping single LTRS item into multiple MML items.

Even at the first sight, among obvious notions from mathematical folklore, two basic objects are especially correlated with our area of research: partitions (resp. coverings in more general approach [24]) and intervals [20]. Querying MML, we discovered that lattices of partitions were formalized already pretty well, but to our big surprise, this was not the case of intervals, so we had to make some preparatory work by ourselves.

Instead of using the ordered pair of approximations, we can claim that both coordinates are just arbitrary objects, so that we can do just a little bit reverse mathematics. It happens even in the heart of rough set theory – correlation of indiscernibility relation properties with those of approximation operators in style of [23] or underlying lattice properties [11] could lead us to develop some general theory of mathematical objects. One of the significant matchings discovered automatically with the help of our work was that classification of rough sets basically coincides with that of objects described by Isomichi – first class objects are precisely crisp sets, second class – rough sets with non-empty boundary, so third class just vanishes in equivalence-based approximation space. Details of this correspondence can be found in [6].

During our project of making both formal approaches to incomplete information accessible in the Mizar Mathematical Library, an extensive framework was created, besides lattices of rough sets (and also of fuzzy sets). Its summary is shown in Table 3.

Table 3 lists our MML articles about rough sets (started with ROUGHS), while the first group contains some preliminary notions and properties needed for smooth further work. We listed only MML items of which we are authors – among nearly 1250 files written by over 250 people. The complete list could contain nearly 20 files containing about 40 thousand lines of Mizar code (so it is about 2% of the MML). Not all of them
Table 3. List of our submissions to MML concerning described topics

<table>
<thead>
<tr>
<th>MML Identifier</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERVA1</td>
<td>algebra of intervals (including lattice of rough sets)</td>
</tr>
<tr>
<td>NELSON_1</td>
<td>Nelson algebras</td>
</tr>
<tr>
<td>YELLOW_1</td>
<td>products of posets</td>
</tr>
<tr>
<td>ROBBINS1</td>
<td>Robbins problem and its solution</td>
</tr>
<tr>
<td>ROBBINS2</td>
<td>correspondence between relational structures and lattices</td>
</tr>
<tr>
<td>ROBBINS3</td>
<td>ortholattices and orthoposets</td>
</tr>
<tr>
<td>SHEFFER1</td>
<td>alternative axiomatization of Boolean lattices</td>
</tr>
<tr>
<td>LATSUM_1</td>
<td>the operation of addition of posets and lattices</td>
</tr>
<tr>
<td>LATTICEA</td>
<td>prime filters and ideals in distributive lattices</td>
</tr>
<tr>
<td>POSET_2</td>
<td>flat posets</td>
</tr>
<tr>
<td>LATTAD_1</td>
<td>almost distributive lattices</td>
</tr>
<tr>
<td>ROUGHS_1</td>
<td>introduction to rough sets</td>
</tr>
<tr>
<td>ROUGHS_2</td>
<td>relational characterization of rough sets</td>
</tr>
<tr>
<td>ROUGHS_3</td>
<td>Zhu’s paper [23] formalized (to appear)</td>
</tr>
<tr>
<td>ROUGHS_4</td>
<td>topological aspects of RST</td>
</tr>
</tbody>
</table>

are tightly connected with the theory of rough sets – they expand the theory of intervals, topologies, relational structures, and – last but not least – lattices, to list more notable areas of mathematics.

Following Grätzer’s classical textbook [10] in Mizar, practically all chapters from Järvinen work are covered, maybe except Section 8 on information systems (reducts are poorly covered in Mizar, but e.g. the Mizar article about Armstrong systems on ordered sets was coauthored by Armstrong himself). Also the final section on lattices of rough sets is not sufficiently covered in the MML, because not all combinations of properties of binary relations were considered. Our personal perspective was that [11] was even better written from the viewpoint of a developer of MML. The fact is that the theory is much simpler than in CCL by Gierz et al. (and it is more self-contained), although a little bit painful – at least from the formal point of view – category theory was used in a similar degree.

6 Conclusions and Future Work

In order to widen the formal framework within the MML, we should construct more algebraic models both for rough and fuzzy sets (as the aforementioned Stone algebras). Although we underlined the possibility of using external provers, we already applied some automatic tools available in the Mizar system aiming at discovering alternative proofs [17] and unnecessary assumptions to generalize the approach (similarly to generalizing from equivalence into tolerance relations in case of rough sets).

It is hard to describe all the formalized work which was done during this project in such a short form; however we tried to outline the very basic constructions. We hope to demonstrate the working prototype of annotated version of LTRS during CS&P
2015 conference, and, of course, to incorporate it into the MML to be explored with the help of Mizar internal tools. We hope to spend about a year of work to finish it completely. Our work was very much inspired by Järvinen [11] and Zhu [23], although regular formalization of this topic started back in 2000, much before both these papers. Some copyright issues should be solved as the best solution is just to publish [11] with links pointing out to appropriate Mizar items and proofs. In the era of expanding open access model for research we hope some limitations can vanish – although some issues connected with the repositories of knowledge formalized with the help of computers are quite interesting [2] and GNU/Creative Commons model is promising.

Mizar code enables also the information exchange with other proof assistants through its XML intermediate format. Even if the Mizar code is relatively well readable, \LaTeX version and HTML script with expandable and fully hyperlinked proofs are available. All Mizar files are also translated into TPTP [18] (first order logic form which can serve as a direct input for Thousands of Problems for Theorem Provers) and this could be one of the fundamental gains for the rough set community.

References

Gained and Excluded Classified Actions by Dynamic Security Policies *

Damas P. Gruska
Institute of Informatics, Comenius University
Mlynska dolina, 842 48 Bratislava, Slovakia
gruska@fmph.uniba.sk

Abstract. Dynamic security policies and formalisms for expressing information on private actions obtained by observing public ones are presented. Two sets of private actions are considered. The set of actions which execution is guaranteed according to observations and the set of actions which execution is excluded according to observations of public actions. Moreover, we consider also intruders which have limited memory capacity to record these sets during an attack as well as policies which change due to elapsing of time.

Key words: dynamic security policy, gained and excludes actions, information flow, security, non-interference

1 Introduction

Information flow based security properties assume an absence of any information flow between private and public systems activities. This means, that systems are considered to be secure if from observations of their public activities no information about private activities can be deduced. Such security properties could be quantified to avoid the unrealistic requirement of absolutely none information flow and hence to express an amount of possibly "leaked secrecy" in several ways. For example, Shannon’s information theory was applied for simple imperative languages (see [CHM07,CMS09]) or for process algebras (see [Gru08]). Another possibility is to exploit probabilistic theory as it was used for process algebras in [Gru09] or by expressing subsets of private actions which occurrence (and not occurrence) can be deduced by an intruder who can observe public behaviour of the system [Gru11]. All above mentioned concepts are based on a static security policy, i.e. policy which is not changed during executions. This approach seems to be rather restrictive for applications where their security policies (classification, declassification etc.) change dynamically during runtime. Hence, there is a growing research and a number of papers devoted to dynamic security policies tailored for various formalizations and computational paradigms. For instance, in the case of imperative programs, security policy requires that values of classified variables could not be obtained by observing public ones, what can be formalizes by an equivalence relation on values of program’s variables. In the case of a dynamic security policy, this relation can change during a computation (see, for example [DHS15]). In general, a

* Work supported by the grant VEGA 1/1333/12.
dynamic security property permits different information flows at different points during program/system’s execution.

The aim of this paper is to formulate dynamic security policies for security concepts based on the sets of gained and excluded private actions [Gru11]. In this case, a dynamic security policy defines a set of private actions at a given state of execution. That means that the set of private actions is not fixed but it can change dynamically during system’s execution. We study the resulting concepts and we show how they are related to traditional static security properties as well. Later, we consider also so called limited intruders (intruders with limited storage to record obtained information) who always try to be prepared for declassifications of private actions but have to take into account also new classification of non-public, so called invisible, actions. We will define also a special class of dynamic security policies which change only with elapsing of time.

The paper is organized as follows. In Section 2 we describe the timed process algebra TPA which will be used as a basic formalism. In Section 3 we formalize sets of gained and excludes private actions with respect to a static security policy. In Section 4 we define gained and excludes private actions with respect to dynamic security policies. The next section is devoted to limited intruders and time dynamic security policies. Section 6 contains discussion and plans for a future work.

2 Timed Process Algebra

In this section we define Timed Process Algebra, TPA for short. TPA is based on Milner’s CCS (see [Mil89]) but the special time action $t$ which expresses elapsing of (discrete) time is added (see also [Gru10]). The presented language is a slight simplification of the Timed Security Process Algebra (tSPA) introduced in [FGM00]. We omit the explicit idling operator $\iota$ used in tSPA and instead of this we allow implicit idling of processes. Hence processes can perform either “enforced idling” by performing $t$ actions which are explicitly expressed in their descriptions or “voluntary idling”. But in the both cases internal communications have priority to action $t$ in the case of the parallel operator. Moreover we do not divide actions into private and public ones as it is in tSPA. TPA differs also from the tCryptoSPA (see [GM04]). TPA does not use value passing and strictly preserves time determinacy in case of choice operator $+$ what is not the case of tCryptoSPA.

To define the language TPA, we first assume a set of atomic action symbols $A$ not containing symbols $\tau$ and $t$, and such that for every $a \in A$ there exists $\overline{a} \in A$ and $\overline{\overline{a}} = a$. We define $Act = A \cup \{\tau\}$, $Act^t = Act \cup \{t\}$. We assume that $a,b,\ldots$ range over $A$, $u,v,\ldots$ range over $Act$, and $x,y,\ldots$ range over $Act^t$. Assume the signature $\Sigma = \bigcup_{n \in \{0,1,2\}} \Sigma_n$, where
\[
\Sigma_0 = \{\text{Nil}\}
\]
\[
\Sigma_1 = \{x. \mid x \in A \cup \{t\}\} \cup \{[S] \mid S \text{ is a relabeling function}\}
\cup \{M \mid M \subseteq A\}
\]
\[
\Sigma_2 = \{[\cdot], +\}
\]

with the agreement to write unary action operators in prefix form, the unary operators \([S]\), \([M]\) in postfix form, and the rest of operators in infix form. Relabeling functions, \(S : \text{Act} \rightarrow \text{Act}\) are such that \(S(a) = S(\bar{a})\) for \(a \in A\), \(S(\tau) = \tau\) and \(S(t) = t\).

The set of TPA terms over the signature \(\Sigma\) is defined by the following BNF notation:

\[
P ::= X \mid op(P_1, P_2, \ldots P_n) \mid \mu X P
\]

where \(X \in Var\), \(Var\) is a set of process variables, \(P, P_1, \ldots P_n\) are TPA terms, \(\mu X -\) is the binding construct, \(op \in \Sigma\).

The set of CCS terms consists of TPA terms without \(t\) action. We will use a usual definition of opened and closed terms where \(\mu X\) is the only binding operator. Closed terms which are \(t\)-guarded (each occurrence of \(X\) is within some subexpression \(t.A\), i.e. between any two \(t\) actions only finitely many non timed actions can be performed) are called TPA processes. Note that \(\text{Nil}\) will be often omitted from processes descriptions and hence, for example, instead of \(a.b.\text{Nil}\) we will write just \(a.b\).

We give a structural operational semantics of terms by means of labeled transition systems. The set of terms represents a set of states, labels are actions from \(\text{Act}\). The transition relation \(\rightarrow\) is a subset of \(\text{TPA} \times \text{Act} \times \text{TPA}\). We write \(P \xrightarrow{x} P'\) instead of \((P, x, P') \in \rightarrow\) and \(P \not\xrightarrow{x}\) if there is no \(P'\) such that \(P \xrightarrow{x} P'\). The meaning of the expression \(P \xrightarrow{x} P'\) is that the term \(P\) can evolve to \(P'\) by performing action \(x\), by \(P \xrightarrow{x}\) we will denote that there exists a term \(P'\) such that \(P \xrightarrow{x} P'\). We define the transition relation as the least relation satisfying the inference rules for CCS (see [Mil89]) plus the following inference rules:

\[
\begin{align*}
\text{Nil} & \xrightarrow{} \text{Nil} & \text{A1} & \mu X P & \xrightarrow{x} \mu X P \\
\text{Pa1} & \frac{P \xrightarrow{x} P', Q \xrightarrow{\tau} Q'}{P | Q \xrightarrow{\tau} P' | Q'} & \text{S} & \frac{P \xrightarrow{\tau} P', Q \xrightarrow{\tau} Q'}{P + Q \xrightarrow{\tau} P' + Q'}
\end{align*}
\]

Here we mention the rules that are new with respect to CCS. Axioms \(A1, A2\) allow arbitrary idling. Concurrent processes can idle only if there is no possibility of an internal communication (\(Pa1\)). A run of time is deterministic (\(S\)). In the definition of the labeled transition system we have used negative premises (see \(Pa1\)). In general this may lead to problems, for example with consistency of the defined system. We avoid these dangers by making derivations of \(\tau\) independent of derivations of \(t\). For an explanation and details see [Gro90]. Regarding behavioral relations we will work with the timed version of weak trace equivalence. Note that here we will use also a concept of observations which contain complete information which includes also \(\tau\) actions and not
just actions from \( A \) and \( t \) action as it is in [FGM00]. For \( s = x_1, x_2, \ldots, x_n, x_i \in Actt \) we write \( P \stackrel{\Delta}{\rightarrow} \) instead of \( P \xrightarrow{x_1, x_2, \ldots, x_n} \). We suppose that all actions are divided into two groups, namely public (low level) actions classified actions (see [Gru11]). We suppose that all actions are divided into two groups, \( \in H, L \cap L = \emptyset \). Moreover, we suppose that \( H \neq \emptyset \) and \( L \neq \emptyset \) and that for every \( h \in H, l \in L \) we have \( \overline{\overline{h}} \in H, \overline{l} \in L \). Hence the division is given by the set of high level actions. To denote sequences of public actions, i.e. sequences consisting of actions from \( L \cup \{t\} \) and sequences of private actions from \( H \), we will use notation \( \overline{l}, \overline{l'}, \ldots \) for sequences from \( \{L \cup \{t\}\}^* \) (note that elapsing of time - i.e. \( t \) action is also a public action) and \( \overline{h}, \overline{h'}, \ldots \) for sequences from \( H^* \), respectively. The set of actions could be divided to more than two subsets, what would correspond into more levels of classification. All the following concepts could be naturally extended to such setting.

First we define a set of private actions which occurrence can be learned by an intruder who see a process to perform a sequence of public actions \( \overline{l} \) (we will call such action as gained actions).

**Definition 3.** Let \( P \in TPA \) and \( \overline{l} \in T_{\text{tr}}(P) \). Then the occurrence of the set of private action which can be gained about \( P \) by public observing \( \overline{l} \) is defined as follows:

\[
g_H(P, \overline{l}) = \{h|h \in H, P \not\Rightarrow_{H \setminus \{h\}} \}.
\]
According to Definition 3 the set of private actions \( g_H (P, \tilde{l}) \) is the one which has to be performed by \( P \) if an intruder sees \( P \) to perform public actions \( \tilde{l} \).

**Example 1.** Let \( P = l_1.h.l_2.Nil + l_1.l_2.Nil \) and \( P' = l_1.h.h'.l_2.Nil + l_1.h.l_2.Nil \). Let \( \tilde{l} = l_1.l_2 \) then we have \( g(P, \tilde{l}) = \emptyset, g(P', \tilde{l}) = \{h\} \).

**Definition 4.** Let \( P \in TPA \). Then the occurrence of the set of private action which can be excluded by observing \( P \) performing public action \( \tilde{l} \) (i.e. \( \tilde{l} \in Tr_t H (P) \)) is defined as follows:

\[
e_H (P, \tilde{l}) = \bigcap_{P \models \preceq M} H \setminus M.
\]

If we have that \( e_H (P, \tilde{l}) = \emptyset \) that means that an intruder after observing \( \tilde{l} \) cannot exclude occurrence of any private action.

There is no direct correlation between sets \( g(P, \tilde{l}) \) and \( e(P, \tilde{l}) \) since there are processes such that for one is the former set empty and the later nonempty and vice versa. If both of them are empty, that means, that an intruder can learn practically nothing on private actions by observing process \( P \) and seeing it to perform \( \tilde{l} \). In some sense \( g(P, \tilde{l}) \) and \( e(P, \tilde{l}) \) are complementary as it is stated in the following proposition (see [Gru11]).

**Proposition 1.** For every process \( P \) and every \( \tilde{l}, \tilde{l} \in Tr_t H (P) \) it holds \( g(P, \tilde{l}) \cap e(P, \tilde{l}) = \emptyset \) and \( \emptyset \subseteq g(P, \tilde{l}) \cup e(P, \tilde{l}) \subseteq H \).

Now we are prepared to formulate how much information can be gained by observing public activities \( O, O \subseteq L^* \) of a process. The formal definition follows.

**Definition 5.** Let \( P \in TPA \). By \( g_H^O (P) \) we will denote the set of private actions which occurrence by \( P \) can be gained (detected) by an intruder observing sequences of public actions from \( O \) as

\[
g_H^O (P) = \bigcup_{\tilde{l} \in O} g_H (P, \tilde{l}).
\]

We say that no private information can be gained by observing \( P \) by \( O \) if \( g_H^O (P) = \emptyset \).

Now we can define a set of private actions which occurrence could be excluded by the set of observations \( O \).

**Definition 6.** Let \( P \in TPA \). Then the occurrence of the set of private action which executions could be excluded by the set of observations \( O, O \subseteq Tr_t H (P) \) is defined as follows:

\[
e_H^O (P) = \bigcup_{\tilde{l} \in O} e_H (P, \tilde{l}).
\]

For a given process \( P \) the size of sets \( g_H^O (P) \), \( e_H^O (P) \) with respect to the size of \( H \) and \( O \), give us another quantification of security. For example, small \( |O| \) and big \( |g_H^O (P)| \) and/or \( |e_H^O (P)| \) indicate a rather low level of security.
4 Dynamic Security Policies

Division of actions to public and private ones is based on fixed (static) security policy which is not changed during system computation. This approach seems to be rather restrictive for applications where their security policies (classification, declassification etc.) change dynamically during runtime. In the presented framework by a policy we mean some set $M$ of actions which are supposed to be private at the given points during program/system’s execution. By dynamic security policy $D$ we mean a partial mapping which assigns to every process $P$ and sequence of actions some policy i.e. subset of actions. Moreover we require that $D$ is uniquely defined with respect to weak bisimulation.

**Definition 7.** By dynamic security policy we mean partial mapping $D : TPA \times Act^* \rightarrow 2^{Act}$ such that $D(P, s) = D(P', s)$ whenever $P \approx P'$.

Hence by $D(P, s)$ we denote the set of actions which are private after the execution of $s$ by $P$ if $P \xrightarrow{s} P'$ otherwise $D(P, s)$ is not defined. Now we can define the set of gained actions with respect to dynamic security policy $D$. In this case an intruder gains private actions given by a security policy valid at the moment and from this set removes actions which are declassified. To do so we divide every execution trace to (maximal) intervals during which a policy is not changed. The formal definition follows.

**Definition 8.** Let $P \in TPA$. Let $s \in Tr(P)$ and $s = s_1 \ldots s_k$ such that $s_i = x_1^i \ldots x_{n_i}^i$ for $i = 1, \ldots, k$. such that $P \xrightarrow{x_1^i} P_{i+1}$ where $P = P_1$ i.e. $P_1 \xrightarrow{x_1^i} P_1^i, \ldots, P_{i+1}$ such that $D(P_i, \epsilon) = H_i$ and also $D(P, x_1^1, \ldots, x_j^i | Act(H_i)) = H_i$ for every $j$, $1 \leq j < n_i$. Moreover, we suppose that $H_i \neq H_{i+1}$. Let $\tilde{l} = \tilde{l}_1, \ldots, \tilde{l}_n$ where $\tilde{l}_i = s_i | Act(H_i)$. Then we define

$$g_D(P, \tilde{l}_1) = g_{H_1}(P, \tilde{l}_1)$$

$$g_D(P, \tilde{l}_1, \ldots, \tilde{l}_{n+1}) = (g_D(P, \tilde{l}_1, \ldots, \tilde{l}_n) \cap H_{i+1}) \cup g_{H_{n+1}}(P_i, \tilde{l}_{n+1}).$$

**Definition 9.** Let $P \in TPA$. By $g_D^O(P)$ we will denote the set of private actions which occurrence by $P$ can be gained (detected) by an intruder observing sequences of public actions from $O$ under dynamic security policy $D$ as

$$g_D^O(P) = \bigcup_{\tilde{l} \in O} g_D(P, \tilde{l}).$$

We say that no private information can be gained by observing $P$ by $O$ if $g_D^O(P) = \emptyset$.

We can define partial ordering between dynamic security policies.

**Definition 10.** Let $D, D'$ are two dynamic security policies. We say that $D$ is stronger than $D'$ (denoted by $D' \preceq D$) if for every $P$ and $s$ it holds $D'(P, s) \subseteq D(P, s)$. 
Both the ordering of dynamic security properties as well as ordering (by inclusion) of sets of observations influence resulting sets of gained action as it is stated by the following proposition.

**Proposition 2.** For every process $P$, sets of observations $O, O'$ and dynamic policies $D, D'$ such that $O \subseteq O'$ and $D' \preceq D$ it holds $g^O_D(P) \subseteq g^{O'}_{D'}(P)$ and $g^O_D(P) \subseteq g^{O'}_{D'}(P)$, respectively.

**Proof.** Sketch. The first part of the proof follows directly from Definition 9. The second part follows from Definitions 8 and 10.

Now we can show how the property "no private information can be gained by observing $P$" by dynamic security policy is related to persistent variant of absence-of-information-flow property, so called Strong Nondeterministic Non-Interference (SNNI, for short). We recall its definition (see [FGM00]). Process $P$ has SNNI property (we will write $P \in SNNI_H$) if $P \setminus H$ behaves like $P$ for which all high level actions are hidden for an observer. To express this hiding we introduce hiding operator $P/M$, for which it holds if $P \xrightarrow{a} P'$ then $P/M \xrightarrow{a} P'/M$ whenever $a \notin M \cup \overline{M}$ and $P/M \xrightarrow{\tau} P'/M$ whenever $a \in M \cup \overline{M}$. Moreover, process has persistent SNNI property (denoted by PSNNI) if also all its successors have SNNI property. Formal definition of SNNI and PSNNI follows.

**Definition 11.** Let $P \in \mathbb{TPA}$. Then $P \in SNNI_H$ iff $P \setminus H \approx_t P/H$ and $P \in PSNNI_H$ iff $P' \in SNNI_H$ for every $P', P' \in \text{Succ}(P)$.

The persistent variant of SNNI property is stronger than SNNI itself as it is expressed by the next proposition.

**Proposition 3.** $PSNNI_H \subset SNNI_H$.

**Proof.** Clearly $PSNNI_H \subseteq SNNI_H$. Let $P = (l.l.Nil + h.e.(h.l.Nil + l.Nil))$. Then it is easy to check that $P \notin PSNNI_H$ but $P \in SNNI_H$.

Now we are ready to formulate relationship between PSNNI property and the set of gained actions under a constant dynamic security policy.

**Proposition 4.** If $P \in PSNNI_H$ then $g^O_{D_H}(P) = \emptyset$ for constant dynamic policy $D_H$ which assigns the set $H$ to every process and sequence of actions.

**Proof.** The main idea. Let $P \in PSNNI_H$ and suppose that $g^O_{D_H}(P) \neq \emptyset$. Hence there exists $P', P' \in \text{Succ}(P)$ and such that and subsequence $o'$ of some observation $o$ from $O$, $o \in Tr_{l_H}(P')$ and $h, h \in H$ and such that $P' \xrightarrow{o'} P'/H \setminus \{h\}$. But then there exists sequence $s$ which contains $o'$ and $h$ such that $s \in Tr_{l}(P'/H)$ but $s \notin Tr_{l}(P'/H \setminus H)$ i.e. $P'/H \notin l P/H$ i.e. $P' \notin SNNI_H$ and hence $P \notin PSNNI_H$.

The inverse of the previous proposition does not hold as it shows the following example.
Example 2. Let \( P = \sum_{1 \leq i \leq n} h_i.\text{Nil} \) and \( H = \{h_1, \ldots, h_n\} \). Then \( g_{DH}^O(P) = \emptyset \) but \( P \) has not PSNNI property since \( P \setminus H \neq \emptyset \). Indeed \( P \setminus H \) cannot perform the sequence of action \( \tau.l \) while \( P/H \) can perform it and an intruder seeing \( l \) can deduce that a private action was performed.

Corollary. Let \( D \preceq D_H \) and \( P \in PSNNI_H \) then \( g_{DH}^O(P) = \emptyset \).

Proof. The proof follows from Proposition 2 and 4.

Now we will define sets of excluded private action first for a single observation and later for a set of observations.

Definition 12. Let \( P \in TPA \). Let \( s \in Tr_t(P) \) and \( s = s_1 \ldots s_k \) such that \( s_i = x_{i1} \ldots x_{in_i} \), for \( i = 1, \ldots, k \) such that \( P_i \xrightarrow{\epsilon} P_{i+1} \) where \( P = P_1 \) i.e. \( P_i \xrightarrow{x_{i1}} P_{i1}, P_i \xrightarrow{x_{i2}} P_{i2}, \ldots, P_i \xrightarrow{x_{in_i}} P_{in_i} \) such that \( D(P_i, \epsilon) = H_i \) and also \( D(P_i, x_{i1} \ldots x_{in_i} | Act_t \setminus H_i) = H_i \) for every \( j, 1 \leq j < n_i \). Moreover, we suppose that \( H_i \neq H_{i+1} \). Let \( \tilde{l} = \tilde{l}_1, \ldots, \tilde{l}_k \) where \( \tilde{l}_i = s_i | Act_t \setminus H_i \). Then we define

\[
e_D(P, \tilde{l}_i) = e_{H_i}(P, \tilde{l}_i) \]
\[
e_D(P, \tilde{l}_1, \ldots, \tilde{l}_{n+1}) = (e_D(P, \tilde{l}_1, \ldots, \tilde{l}_n) \cap H_{i+1}) \cup e_{H_{i+1}}(P, \tilde{l}_{n+1}).
\]

Definition 13. Let \( P \in TPA \). By \( g^D(P) \) we will denote the set of private actions which occurrence by \( P \) can be excluded by an intruder observing sequences of public actions from \( O \) under dynamic security policy \( D \) as

\[
e_D^O(P) = \bigcup_{l \in O} e_D(P, \tilde{l}).
\]

We say that no private information can be excluded by observing \( P \) by \( O \) if \( g_{DH}^O(P) = \emptyset \).

For excluded action we can formulate the similar property which holds for gained actions. Also the proof is similar.

Proposition 5. For every \( P' \) and \( D', \) \( D' \preceq D \) it holds \( e^O_{D'}(P') \subseteq e^O_D(P) \).

There is no direct correlation between sets \( g_D(P, \tilde{l}) \) and \( e_D(P, \tilde{l}) \) since there are processes such that the former set is empty and the later one is nonempty and vice versa. If the both of them are empty, that means, that an intruder can learn practically nothing on private actions by observing process \( P \) to perform \( \tilde{l} \) under the dynamic security policy \( D \). In some sense \( g_D(P, \tilde{l}) \) and \( e_D(P, \tilde{l}) \) are complementary as it is stated in the following proposition.

Proposition 6. For every process \( P \) it holds \( g_D(P, \tilde{l}) \cap e_D(P, \tilde{l}) = \emptyset \) and \( \emptyset \subseteq g_D(P, \tilde{l}) \cup e_D(P, \tilde{l}) \subseteq \bigcup_{s, s' \subseteq D(P, s)} D(P, s) \).
Proof. Let \( h \in g_D(P, \tilde{l}) \). We now that every execution of sequence of visible action \( \tilde{l} \) has to contain \( h \) for some \( P' \in \text{Succ}(P) \) i.e. if \( P' \xrightarrow{\tilde{l}}_{M} \) then \( h \in H_i \). That means \( h \notin H_i \setminus M \) i.e. \( h \notin e_D(P, \tilde{l}) \). As regards the second part of the proposition let us consider process \( P = l.Nil + h.l.Nil \). We have \( g(P, l) = e(P, l) = \emptyset \). If we consider \( H = \{ h \} \) then we see that \( g_D(H, \tilde{l}) \cup D_H(P, \tilde{l}) = H \) i.e. \( \subseteq \) cannot be replaced by \( \subset \) in general.

We could further quantify levels of security by relating size of \( g_D(P) \), \( e_D(P) \) to the size of \( O \) and \( D \) as it was suggested at the end of the previous section.

5 Limited Intruders and Variants of Dynamic Security Policies

In this section we will assume intruders which are aware of dynamicity of security policy and try to learn as much as can be done with limited amount of memory where obtained information can be recorded. That means, (s)he tries to record also invisible but declassified actions for the case that they become classified in the future. First, let us consider three types of actions. Low level (public) actions, which are always visible, and the rest of the actions is called invisible (\( I \)). Moreover, the invisible actions could contain private (high level) actions, i.e. \( A = L \cup I, L \cap I = \emptyset \) and \( H \subseteq I \). Now we will consider dynamic security policies for which \( D(P, s) \subseteq I \). That means that neither the set of visible nor invisible actions are changed by \( D \). Only the set of high level actions can be changed but they are always invisible. We assume that \( |I| \geq n \).

We suppose intruders who try to learn information about all classified actions but also about declassified actions which are not visible at the moment for the case that they will become classified in the future. Moreover, we assume that (s)he has only limited storage to record obtain information and hence in the case that the amount of this information is bigger then memory capacity, then some part of this already obtained information has to be forgotten. To model this we define a mapping which assigns to given set of classified actions \( M \), capacity of storage \( n \) and given set \( N \) a set of subsets of \( N \) with size \( n \) (or less if the capacity of storage is not reached) preferably containing actions from \( M \). The formal definition follows.

**Definition 14.** \( F^n_M(N) = \{ N' \mid \text{where } N' = N \text{ if } |N| \leq n \text{ otherwise if } |N \cap M| \geq n \text{ then } N' \subseteq N \cap M \text{ such that } |N'| = n \text{ and if } |N \cap M| < n \text{ then } N' \subseteq N \text{ such that } N \cup M \subseteq N' \text{ and } |N'| = n \} \).

Mapping \( F^n_M \) could be naturally extended for sets of sets. Let \( T \subseteq 2^I \), then \( F^n_M(T) = \bigcup_{N \in T} F^n_M(N) \). Now we reformulate Definitions 3, 8 and 9 for limited intruders.

**Definition 15.** Let \( P \in TPA \) and \( \tilde{l} \in Tr_{\tilde{l}}(P) \). Then the occurrence of the set of private action which can be gained about \( P \) by public observing \( \tilde{l} \) is defined as follows:

\[
g_H(P, \tilde{l}, n) = F^n_H(\{ h | h \in I, P \xrightarrow{\tilde{l}}_{I \setminus \{h\}} \})
\]
Definition 16. Let $P \in TPA$. Let $s \in Tr_t(P)$ and $s = s_1,\ldots,s_k$ such that $s_1 = x_1^1,\ldots,x^i_1$ for $i = 1,\ldots,k.$ such that $P_i \rightarrow s_i P_{i+1}$ where $P = P_1$ i.e. 
$P_1 \rightarrow x_1^1 P_2^1, P_1 \rightarrow x_2^2 P_2^2, \ldots, P_{n-1} \rightarrow x_n^{n-1} P_{n+1}$ such that $D(P_i,\epsilon) = H_i$ and also $D(P_i, x_1^1,\ldots,x_n^{n-1}) = H_i$ for every $j$, $1 \leq j < n_i$. Moreover, we suppose that $H_i \neq H_{i+1}$. Let $\bar{i} = i_1,\ldots,i_k$ where $i_i = s_1|_{Act(H_i)}$. Then we define

\[ g_D(P,\bar{i},n) = g_{H_1}(P,\bar{i}) \]
\[ g_D(P,\bar{i},\ldots,\bar{i}_{n+1},n) = \mathcal{F}_{H_{i+1}}^\prime((g_D(P,\bar{i},\ldots,\bar{i}_{n}) \cap H_{i+1}) \cup g_{H_{i+1}}(P,\bar{i}_{n+1})). \]

Definition 17. Let $P \in TPA$. By $g_D^O(P)$ we will denote the set of private actions which occurrence by $P$ can be gained (detected) by an intruder observing sequences of public actions from $O$ under dynamic security policy $D$ as

\[ g_D^O(P,n) = \bigcup_{i \in O} g_D(P,\bar{i},n). \]

We say that no private information can be gained by observing $P$ by $O$ if $g_D^O(P) = \emptyset$.

Note that in the previous definition we do not apply $\mathcal{F}_{M}^n(H)$ since the whole concept is based on one time attacks. If $M \in g_D^O(P,n)$ than (there is a possibility that) an intruder can gain actions from $M$. Hence $E = \bigcup_{M \in g_D^O(P,n)} M$ is the set of possibly gained actions. Clearly, with a bigger memory an intruder can learn more as it is stated by the following proposition.

Proposition 7. Let $n \leq m$. Then for every $M$, $M \in g_D^O(P,n)$ there exists $M'$, $M' \in g_D^O(P,m)$ such that $M \subseteq M'$.

Proof. The main idea. It is clear from Definition 14 that mapping $\mathcal{F}_{M}^n$ is monotonic with respect to parameter $n$.

If $n$ sufficiently large limited and unlimited intruders can learn the same as it is stated by the following proposition.

Proposition 8. Let $|\bigcup_s D(P,s)| \leq n$ for every $P$ and $s$. Then $g_D^O(P,n) = g_D^O(P)$.

Proof. The main idea. From Definition 14 we see that $\mathcal{F}_{M}^n(H)$ is identical function if $|\bigcup_s D(P,s)| \leq n$

For $|\bigcup_s D(P,s)| > n$ we cannot say, in general, whether $g_D^O(P,n)$ is equal to $g_D^O(P)$ or not. We cannot say this even if $|D(P,s)| > n$ for some $s$. We could define also sets of excluded actions for limited intruders in the similar way as it is done for gained actions. But instead of that we concentrate on special cases of dynamic security policies. First, we define finite dynamic security policies and then time dynamic security policies.
Definition 18. We say that dynamic security policy is finite iff there exists set \( F = \{ H_1, \ldots, H_m \} \) such that for every \( P \) and \( s \) we have \( D(P, s) \in F \).

Definition 19. By time dynamic security policy we dynamic security policy such that for every \( P \) and \( s, s' \) such that \( s|_{\{t\}} = s'|_{\{t\}} \) it holds \( D(P, s) = D(P, s') \).

In the case of finite dynamic security policy \( D \) we could design process \( P_D \) which could "compute" the both sets of excluded and private actions for finite state processes in the following sense: \( h.g \in Tr_t(C[P]|P_D) \) iff \( h \in e_{O_D}^P(P, n) \) and \( h.e \in Tr_t(C[P]|P_D) \) iff \( h \in g_{O_D}^P(P, n) \) where \( C[\cdot] \) is an appropriate process context (see [Gru10]). In this case we obtain decidability of sets \( g_{O_D}^P(P) \) and \( e_{O_D}^P(P) \) (and their derivatives) for finite state processes which are undecidable in general.

Time dynamic security policies are useful for formalization of systems which allow partial classification/declassification of actions within some time windows. Study of these two specific dynamic security policies we leave for the further research.

6 Conclusions

We have presented several security concepts based on an information flow and dynamic security policies. They express which set of private actions was performed (gained sets) or which set of private actions could be excluded by an intruder observing systems public actions (excluded sets), taking into account dynamic security policies which can change sets of high level actions during runtime. The concepts offer a finer security notion with respect to traditional ones which usually only ensure that an intruder cannot learn that some private action was performed (for example, persistent variant of SNNI). Moreover, the sets of excluded and gained actions can be used for reduction of a space of possible private actions and if the reduction is significant then it really threatens systems security.

Concepts of gained and excluded private actions are complementary. Roughly speaking, only systems for which both the sets - gained and excluded private actions are empty could be considered fully secure with respect to a given dynamic security policy \( D \) and set of observations \( O \). But since this is a very rare situation we have suggested how to numerically express corresponding level of security by relating size of sets of gained or excluded actions to the set of all appropriate actions and the size of \( O \). That means, if the resulting measure is small enough the system can still be considered secure with respect to some given requirements. Later we have introduced the concept of limited intruders, i.e. intruders who have limited storage to record information obtained by observing public system activities. Such intruders could try to record also declassified but invisible action for the case that during execution they could become classified. We had to resolve the case when not all information on invisible actions could be recorded due to the lack of memory space which has an intruder at disposal.

In the future we plan, besides already mentioned research, to investigate also additional covert channels which could be exploited by an intruder. Particularly interesting are termination and divergence channels. They can be exploited by an intruder who can learn that the system is still working but does not react (for example, by power consumption). It might happen, for example, that the system is completely secure if an intruder cannot see termination (or divergence) and vice versa.
References


Designing Reliable Communication for Heterogeneous Computer Systems

Miroslaw Hajder, Janusz Kolbusz, and Roman Korostenskyi
University of Information Technology and Management in Rzeszow
miroslaw.hajder@gmail.com, {jkolbusz, rkorostenskyi}@wsiz.rzeszow.pl

Abstract. This study describes the network design solution to the problem of connecting heterogeneous computer systems based on analysis of multipartite hypergraphs. To do this proposes a mathematical model of reliability for the two modes of operation of the system: with redundancy communication subsystem and the division of communication load. As the evaluation criteria applied solutions expected changes in processing capacity, latency communication and system reliability. Solution design task is sought in the collection Pareto optima, which describes a method for selecting a particular solution in case of equivalence with respect to the vector of the objective function.

Key words: Multipartite hypergraphs, architecture connections, system reliability model, design methodology

1 Introduction

Another important feature of modern processing systems is the variety of offered services. Nowadays, in the same network they are different, often incompatible with communication services (eg. Isochronous and synchronous transfer). This issue requires a change of quality of provided services, through the dynamic allocation of independent communication channels to users or services present in the network specifically in particular, this applies to multimedia services rendered in real-time or services comprising critical infrastructure. Modern distributed systems are also characterized by high dynamics of changes in operating parameters. Load elements of computing and communications is changing rapidly, which prevents the design and execution of the network to meet even medium-term requirements of the users. Another disadvantage occurring in communication subsystems ubiquity of traffic is bursty traffic, obstructing, and sometimes preventing proper functioning of the network. Currently, the solution to the above problems is the use of load leveling, both communications and computing. Moreover, an effective solution to most of these problems can be also providing a flexible reconfiguration of connections, preferably at the logical level, without having to modify the hardware architecture. In this way, links can be dynamically adapted to the current traffic pattern.

Effective methods of reconfiguring connections should be seen in the use of modern communication technologies, especially those that allow to realize it at the logical level and which additionally improve the utilization of physical communication channels. An interesting issue is the construction of multi-channel network of bus-sharing bus
logic dynamic range conversion of a set of buses to which the user is attached. Because of that it becomes possible to adapt to the existing network architecture in the traffic patterns. However, the use of such architecture requires solving a specific design task, which for obvious reasons should be characterized by an acceptable time complexity and memory. Because of the combinatorial nature of the task it is difficult to meet.

These issue may include the design task to build large class of system configuration tasks. This task is mostly decomposed into three basic subtasks: a. The selection of system components; b. their deployment; c. determining the connection between them. In previous work, component selection subtask is solved inter alia by implemented using for this purpose methods seeking the shortest path [1], [2] Backpack block [2], [3] the clustering multipartite graph [4], mullioned clique [5], morphological analysis. Subsequently, a solution subtasks arrangement of the components, currently is the most frequently used variants task assignment [2], [6], [7], [8]. To determine the connections between system components there is the most commonly used a method of agglomeration [9], [10] and the method solving the task of building the optimal hierarchy [11].

2 Architecture Connections and System Reliability Model

The Fig. 1 shows a distributed system consisting of $K_N$ node calculation - decoration, each of which has a $K_l$ retunes elements of the transceiver and the $K_B$ communication buses. Buses $B_i (i = 1, \ldots, K_B)$ are logical and can be implemented on the basis of methods of reproduction of wave-go in one physical bus $D_F$. The addition of logical channels to any node $N_i$ physical bus $B_F$ is done by using a physical connection channels $l$ and distributors of bus channels $C$.

![Fig. 1. Trunk generalized computing system architecture](image)

Interconnect architecture of the analyzed system can be dynamically reconfigured by tunable elements transmitter - receiver. If you change the traffic pattern elements of the transmitter - receiver extent of the wave changes, and indeed attach themselves
to another logical bus. The system can operate in two modes: a. redundancy communication subsystem; b. the communication load sharing. To evaluate the operating characteristics of the system in one of these modes, the probabilistic model suggested combinatorial, and in particular evaluating the reliability $R$. In this model, the kit: the transceiver - receiving ($\approx$), the physical connection cable ($l$), and a manifold physical channel ($C$) are treated as a single device connection. Let $p_{io}$ - the probability of performance transceiver - receiving, $p_t$ - probability of physical fitness connecting channel; $p_c$ - the probability of the distributor channel efficiency Trunk, the $p_{jk}$ - probability of physical fitness BUS channel. Then, the probability of $p_{ku}$ efficiency of connecting the node to the logical channel BUS is defined as: $p_{ku} = p_{io}p_{sp}p_c$, and $p_{aku}$ probability of the merger selected node with other nodes calculation is equal to $p_{aku} = p_{io}p_{sp}p_c p_{jk}$.

Trunk consider the reliability of a distributed system with connections complete (each compute node is connected to each bus logic) and equal rights computational nodes, which is the most general example of this class of systems. The probability $p_{wue}(k_{wue})$ efficiency connection sets providing connection to the bus channel not less than $k_{wue}^{\min}$ compute nodes with their total number $K_N$ determined by the expression:

$$p_{wue}(k_{wue}) = p_{jk}\sum_{i=k_{wue}^{\min}}^{K_N} C_k^i p_{ku}^i (1 - p_{ku})^{k_{wue} - i} \quad (1)$$

Let $K_B$ to be the amount of bus logic (ie. The maximum multiplicity system interface), $p_{wue}^{sp}$ - likely performance computing node. Then, using expression (1), according to the proposed condition for the system in redundancy, reliability $R$ is defined by the following formula:

$$R = \sum_{j=k_{wue}^{\min}}^{K_N} C_{j}^{K_N} (p_{wue}^{sp})^j (1 - p_{wue}^{sp})^{K_N - j} \sum_{k=1}^{K_B} C_{K_B}^k p_{wue} (j)^k (1 - p_{wue} (j))^{K_B - k} \quad (2)$$

Consider the current system of equal rights computational nodes working in load sharing mode of communication. Let $W(k_{wue}, k_m, \sigma)$ to be the amount of system efficiency states consisting of $k_{wue}$ compute nodes, connected with $k_m$ canals system, which can be selected with the existence of $\sigma$ denials transceiver components. In order to determine the amount of $W(k_{wue}, k_m, \sigma)$ state performance of the system in the event of damage, etc., it is proposed to use the methods of include - exemption. $H_1(\sigma)$ number of states malfunction of the entire system in case of refusal not less than one minimum section for a system with equal rights nodes is equal to:

$$H_1(\sigma) = K_N C_{(K_N - 1)K_B}^{\sigma - K_B} + C_{K_N}^{2} C_{(K_N - 1)K_B}^{\sigma - K_B} \sum_{i=1}^{K_B - 1} C_{K_B}^i = C_{(K_N - 1)K_B}^{\sigma - K_B} K_N + C_{K_N}^{2} \sum_{\alpha=1}^{K_B - 1} C_{K_B}^\alpha \quad (3)$$

Then, the value of $W$ is equal to $W(k_{wue}, k_m, \sigma) = C_{k_{wue}k_m}^{\sigma} + \sum_{i=1}^{i_\sigma} (-1)^i H_i(\sigma)$, where: $C_{k_{wue}k_m}^{\sigma}$ - total number of states $\sigma$ denials system for transmitting elements - receiving; $i_\sigma$ - the maximum number taken into account when assessing the minimum cross sections; $H_i(\sigma)$ - the number of states of a computing system failure, refusal
to transmit elements - receiving no less than the minimum $i$ sections. The probability
$P_{ku}(k_{we}, k_{m}, \sigma)$, $k_{wu}$ ensures consistency nodes by $k_{m}$ bus for refusals $\sigma$ is defined
by the expression:

$$
P_{ku}(k_{we}, k_{m}, \sigma) = W(k_{we}, k_{m}, \sigma) \cdot p_{ku}^{k_{wu}k_{m} \sigma} (1 - p_{ku})^\sigma \quad (4)$$

Using the expression (4), the likelihood of $P_{ku}(k_{we}, k_{m})$ ensures the consistency
of computing nodes $k_{we}$, $k_{m}$ buses, with refusals the existence of $\sigma$ can be written as:

$$
P_{ku}(k_{we}, k_{m}) = \sum_{\sigma=0}^{(k_{we}-1)k_{m}} P_{ku}(k_{we}, k_{m}, \sigma) \quad (5)$$

Using the expression (5), it will determine the likelihood of consistency $P_{uku}(k_{we})$,
$k_{we}$ compute nodes:

$$
P_{uku}(k_{we}) = \sum_{l=0}^{K_{m}} C_{K_{m}}^{l} k_{w}p_{fu}^{l} K_{m}^{l} K_{m}^{-l} p_{ku}(k_{we}, l),$$

where: $K_{m}$ - the minimum required number of buses needed to provide the required
bandwidth. In this way, the reliability of calculation system is equal to:

$$
R = \sum_{n=K_{we}}^{K_{N}} C_{K_{we}}^{s} (p_{we}^{s})^{n} (1 - p_{we}^{s})^{K_{N} - n} P_{uku}(n) \quad (6)
$$

For computing system client-server mode redundancy it will determine the likelihood of $P_{ks}(K_{K}, K_{S})$ efficiency BUS communication channel to which, through the operational elements transmitter - receiver including no less than $k_{m}$ customers with
their total number of $K_{K}$ and $k_{m}$ servers with the total number of $K_{S}$:

$$
P_{ks}(K_{K}, K_{S}) = p_{fu} \sum_{i=k_{m}}^{K_{m}} \sum_{j=k_{m}}^{K_{m}} \cdot C_{K_{m}}^{i} K_{m}^{j} K_{m}^{-j} (1 - p_{ku})^{K_{K} - i} \cdot$$

$$
(1 - p_{ku})^{K_{S} - j} \quad (7)
$$

Let $p_{fu}^{s}$ and $p_{ku}^{s}$ be the likelihood of efficiency for clients and servers nodes, respectively. Then, using the expression (7), reliability $R$ can be written as:

$$
R = \sum_{m=k_{m}}^{K_{m}} C_{K_{m}}^{m} (p_{fu}^{s})^{m} (1 - p_{fu}^{s})^{K_{K} - m} \cdot$$

$$
\sum_{n=k_{m}}^{K_{m}} C_{K_{m}}^{n} (p_{ku}^{s})^{n} (1 - p_{ku}^{s})^{K_{S} - n} \cdot$$

$$\sum_{i=1}^{K_{S}} C_{K_{m}}^{i} (p_{ks}(K_{K}, K_{S}))^{i} (1 - P_{ks}(K_{K}, K_{S}))^{K_{m} - i} \quad (8)$$

Let’s consider the reliability of client-server system with a complete blend
of computing and communication subsystem working in load sharing mode. Let $W(k_{s}, k_{k}, k_{m}, \sigma)$ to be the number of states efficient system consisting of servers $k_{s}$, $k_{k}$clients, $k_{m}$bus, in the presence of $\sigma$ denials transceiver components. For the client-server system, the number of failure conditions the $H_{1}(\sigma)$ no less than a minimum cross section can be determined using the expression:
\[ H_1(\sigma) = (k_k + k_s) C_{k_m}^{\sigma-k_m} + k_k k_s C_{k_m}^{\sigma-k_m} = C_{k_m}^{\sigma-k_m} \left( k_k + k_s + k_k k_s \sum_{\alpha=1}^{k_m-1} C_{\alpha} \right), \]  

(9)

a number of states proper functioning as:

\[ W(k_k, k_s, k_m, \sigma) = C_{k_m(k_k+k_s)}^{\sigma} + \sum_{i=1}^{k_m} (-1)^i H_i(\sigma), \]  

(10)

Where: \( C_{k_m(k_k+k_s)}^{\sigma} \) - the total number of computing system states that may occur at \( \sigma \) refusals. The probability \( P_{ku}(k_s, k_k, k_m, \sigma) \) ensures consistency \( k_s \) servers and \( k_k \) clients using \( k_m \) bus in case of refusal elements transmitter - receiver can be written as:

\[ P_{ku}(k_s, k_k, k_m, \sigma) = W(k_s, k_k, k_m, \sigma) p^{(k_s+k_k)k_m-\sigma} (1 - p_{ku})^\sigma. \]  

(11)

The probability of \( P_{ku}(k_s, k_k, k_m) \) servers to ensure coherence \( k_s \) and \( k_k \) clients using \( k_m \) bus in the event of a refusal elements transmitter - receiver has been defined as:

\[ P_{ku}(k_s, k_k, k_m) = \sum_{s=0}^{(k_s+k_k)k_m} P_{ku}(k_s, k_k, k_m, \sigma), \]  

(12)

and the probability \( P_{ku}(k_s, k_k, k_m) \) of consistency \( k_s \) servers and \( k_k \) clients as:

\[ P_{ku}(k_s, k_k) = \sum_{k_m=1}^{K_m} C_{K_m}^{k_m} p_{f_k}^{k_m} (1 - p_{f_k})^{K_m-k_m} P_{ku}(k_s, k_k, k_m). \]  

(13)

Using the expression (11), (12), and (13) the sought reliability \( R \) will be written as:

\[ R = \sum_{k_m=k_{m_1}}^{K_m} C_{K_m}^{k_m} (p_{f_k})^k (1 - p_{f_k})^{k_m-k} \sum_{s=k_m}^{K_s} C_{K_s}^{s} (p_{s_2})^s (1 - p_{s_2})^{K_s-s} P_{ku}(s, k) \]  

(14)

The above-described methodology we will use for further connections to network design measuring system.

### 3 Task Design and Its Solution

We will consider hypergraph \( H = (V, E) \) comprising a set \( V = \{ v \} \) of vertices and a set of \( E = \{ e \} \) of edges, which represent a subset of the set \( V \), i.e. \( e \subseteq V \). Hypergraph \( H \) is a \( k \)-regular if each of its edge \( e \in E \) consists of \( k \) vertices. On the other hand, hypergraph \( H \) is \( l \)-partite graph, if the set of its vertices is divided into \( l \) subsets \( V_1, V_2, \ldots, V_l \), in such a manner that the vertices of each of the edges \( e \in E \) belong to different parts of the graph, i.e. \( v_i \in V_i \), where \( i = 1, \ldots, l \). For the determination of \( l \)-partite hypergraphs we will use record form \( H = (V_1, V_2, \ldots, V_l) \).

Let’s consider the \( l \)-partite hypergraph \( H = (V_1, V_2, \ldots, V_l) \). In this graph, the part \( a = (V_1^A, \ldots, V_l^A, E_A) \), for \( i = 1, \ldots, l \) and \( V_i^A \subseteq V_i \), where any two edges \( e_1, e_2 \in E_A \) overlap in one and the same vertex \( v \in V_i^A \) and do not overlap at any vertex \( v \in V_i^A \), will be called star. This means that the cardinality of \( V_i^A \) is 1, and
the vertex \( v \in V_1 \), will be called the center of the star. We distinguish the simple and complex stars. If any pair of edges \( e_1, e_2 \in E_A \) covers only in one vertex \( v \in V_1 \), then the star is called simple. Otherwise, a star will be called complex. The number of edges of the star will be called degree. For the edge \( e = (v_1, v_2, \ldots, v_l) \in E \) of the star vertices \( v_1 \) and \( v_2 \) we will call end. In turn, the vertices \( v_2, \ldots, v_{l-1} \) will be determined as internal. Vertices set of the part of graph \( V_2, \ldots, V_{l-1} \) are composed of empty pairs of disjoint sets \( V_i(v_j), v_j \in V_j \), where: \( i = 2, \ldots, l-1, j = i + 1 \).

For hypergraph \( H = (V, E) \) its subhypergraph \( H_1 = (W, U) \) we’ll be called hypergraph for which set of the vertices \( W \) is the vertices subset \( V \) of hypergraph \( H \), i.e. \( W \subseteq V \) and the edge set \( U \) is the edges subset \( E \) of the hypergraph \( H \), wherein if the \( (x, y) \in E \) and \( x, y \in W \), then \( (x, y) \in U \). Hypergraph cohesion component will be called the set of its vertices, such as any of two of its elements there is a path between them, but there is no path leading from the vertex of belonging to this collection to any other vertex outside. If there is in the subhypergraph \( H_1 = (V_1, E_1) \) of hypergraph \( H \) consistency of each component there is a star with center at some vertex \( v \in V_1 \), the \( H_1 \) we will call the coverage of hypergraph \( H \) stars.

Fixed design task was to find such a connection structure that will ensure the maximization or minimization of operating parameters, such as communication delay, errors in access to the communication medium as a result of his occupation, performance computational processing nodes and others. Such a task can be solved by seeking the cover at least the stars of trigeminal graph. Let’s consider the task.

**Input data.** As an input data we use the design task:

1. \( B = \{ b \} \) - A set of logical communication bus dedicated by physical channel under communication using any of the methods of reproduction communication;
2. \( F = \{ f \} \) - A set of access protocols, which describes the functioning logical BUS communication [12];
3. \( N = \{ n \} \) - A set of customers of the system, using BUS communication channels.

Customers are divided into groups \( d \in D \) taking their communication requirements into account, where \( D = \{ d \} \) - is a set of types of communication requirements. Elements of the set \( D \) shall be as follows: \( d = 0 \) - service streams sensitive to errors and latency; \( d = 1 \) - support for latency-sensitive flows; \( d = 2 \) - service streams are sensitive to transmission errors; \( d = 3 \) - handling sensitive streams not being sensitive to these factors.

**The definition of design task.** Each of the compute nodes \( n \in N \) should be assigned a set of \( M \) bus \( b \), which is a subset of \( B \), i.e. \( M \subseteq B \), each of which will operate on the basis of one of the access protocols \( f \in F \).

**The mathematical model.** The task design is iteratively solved. In each of the steps and for each of the compute nodes there is sought one bus \( b \in B \) providing for node communication services using Access Protocol \( f \in F \). To avoid multiple of includes a node on the same bus at each successive step from the available buses the client is excluded from those for whom it has already been connected.

The mathematical model is based on the 3-partite hypergraph \( H = (V, E) = (X, Y, Z, E) \). Busses from the set \( B = \{ b \} \) correspond to the vertices of the first part \( x \) \((x \in X)\). Each of them (at the same time each logical bus) is assigned to the label \( \eta (x) \)
determining the transmission characteristics of the bus, in the simplest case, the number of nodes measured, that it can handle.

The \( f \) elements of the set \( F \) bus access protocols correspond to the vertices \( y \) of the second part of hypergraph \( H (y \in Y) \), and the elements \( n \) from \( N \) compute nodes correspond to the vertices \( z \) of the third part of hypergraph \( (z \in Z) \). The set of edges \( E = \{ e \} \) includes all three vertices \( (x, y, z) \) such that \( x \in X, y \in Y, z \in Z \). There are permitted only those edges, for which a selected bus the client can handle \( b_i \in X \) using a communication protocol \( f_l \in Y \). Collection \( E = \{ e \} \) of all edges is determined essentially by a set of all admissible triples \( e = (x, y, z) \). Taking account of the value of the parameter \( n (x) \) for \( x \in X \) in hypergraph \( H = (V, E) = (X, Y, Z, E) \) permissible step design task solution will be any of its sub hypergraph \( \beta = (V_\beta, E_\beta) \) for \( V_\beta \subseteq V \) and \( E_\beta \subseteq E \) of which each component represents the simple consistency star of stage with the center in the apex \( x \in X \). As \( S = S (H) = \{ s \} \) we denote the set of all feasible solutions of tasks covering hypergraph \( H \) stars.

Each of edges \( e \in E \) of hypergraph \( H = (V, E) \) there is assigned three scales describing the following characteristics solutions:

1. \( \omega (e) = \phi (x, y, z) \) - Expected customer conversion processing performance in a system in which the client is supported in communication with the bus \( X \), which uses a communication protocol \( y \). To evaluate the performance characteristics of the proposed process there were used in [13], [14], [15]. Described measures therein are modified so that they reflect the change in performance which are due to changes in the architecture of calls. Because of the nature of the bus network connecting the primary measure of performance is the number of nodes, for which there is available transportation network [16].

2. \( \xi (e) = \phi (x, y, z) \) - Expected change in the communication delay on demand of the customer for these conditions. The level of changes in the delay is determined on the basis of the stochastic model using the method described in [17].

3. \( \psi (e) = \phi (x, y, z) \) - Expected change in system reliability for client-server preserved the conditions of point 1. To determine the reliability of the method was applied changes presented in §2.

**Solution design task.** The rating of the solutions will be shown as a multi-criteria. The proposed set of criteria is obviously exemplary, and his selection depends on the needs of the designer, in particular concerning the nature of the future operation of the network merger. For the case of under consideration there are the three functions described below.

Let’s consider the set \( A = \{ a \} \) of acceptable solutions of design task. For each of them we define the following characteristics assessing the quality of solutions:

1. Criterion performance computing: \( \Phi_1 (a) = \max_{a \in A} \min_{e \in E_a} \omega (e) \), where: \( E_a \) - sub of edge of hypergraph \( H \) belonging to solution \( a \). Using this criterion, we strive to maximize the minimum level of performance (computing or communication) system.

2. Communication delay criterion: \( \Phi_2 (a) = \min_{e \in E_a} \xi (e) \), which provides network search junction with minimal delay summary. For systems with varying levels
of validity of nodes, the value $\xi(e)$ of expected changes in the communication delay is called using vertex priority.

3. The criterion of reliability: $\Phi_3(a) = \max \sum_{e \in E_3} \psi(e)$. This criterion provides a network architecture search for which the total reliability is maximum likewise in the case of a delay criterion communication $\Phi_2$.

The possibilities of the above method is not limited to the application of the summation as a $\min$ or $\max$. To assess the quality of solutions it can be used any of the method of folding (convolution) parameters, including methods taking the weight of each sub-parameters into account. These sub-criteria are related by a function to form $\Phi(a) = (\Phi_1(a), \Phi_2(a), \Phi_3(a))$. Multi-criteria objective function $\Phi(a)$ defines a set $A$ feasible solutions, a set of Pareto $A^P$ composed of Pareto optima $a^P$. If two solutions $a_1, a_2 \in A$ vector objective function $\Phi(a)$ are equivalent, then the set of $A^P$ is secreted full set of alternatives $A^A$, which is, in fact, a maximum system vectorially different optima Pareto.

4 The Research, Results and Further Work

The work approach used to create a multi-channel design methodology destined for fieldbus communication service systems, client-server computing. The existing methodology is focused on providing definite level of computing capacity of the entire system, regardless of its reliability parameters. In the presented in the work version, methodology seeks the optimal solution with respect to multi-criteria objective function that one of the criteria is reliability. Depending on how sub-criteria ties and a set of restrictions proposed methodology also allows you to specify the connection architecture, characterized by: a. maximum reliability with certain: the minimum efficiency and maximum delay network communication links; b. the minimum communication delay with the reduction in the minimum reliability and performance; c. maximum computing performance of a specified acceptable level of reliability and delays. There was also tested solution, the aim of which was to design load leveling various communication buses. For each of the solutions sought there are limited the maximum cost of construction. We analyzed the network of connections complete and partial, flat and hierarchical.

Simulation studies of obtained architectures for computing model client-server based on the methodology presented in [16] showed that the use of multi-channel communication systems can flexibly adapt to the current needs of the computing system. Increasing performance computing system with unchanging resources, obtained by re-configuring its connections reached 260%. Deviations in terms of the burden of communication channels does not exceed 31% and the probability of rejecting a service request has fallen nearly 8-fold. The algorithm of the interconnection network is characterized by polynomial time complexity, which can react in real time to any change in traffic patterns.

Further research will focus on finding effective methods of searching for coverage of celebrities simple multipartite hypergraph, which will allow the use of graphs in the design process of any of valor, and this will introduce further design criteria.
References

Knowledge Pit - A Data Challenge Platform*

Andrzej Janusz¹, Dominik Slezak¹,², Sebastian Stawicki¹,², and Mariusz Rosiak

¹ Institute of Mathematics, University of Warsaw, Banacha 2, 02-097, Warsaw, Poland
² Infobright Inc., Poland

{janusza,slezak,stawicki}@mimuw.edu.pl
mariusz.rosiak@gmail.com
http://www.dominikslezak.org

Abstract. Knowledge Pit (https://knowledgepit.fedcsis.org) is a web platform created to facilitate organization of data mining competitions. Its main aim is to stimulate collaborative research for solving practical problems related to real-life applications of predictive analysis and decision support systems. What makes Knowledge Pit different from other data challenge platforms is the fact that it is a non-commercial project focusing on a collaboration with international conferences. It promotes the idea of open research and encourages young researchers to involve in projects related to data science. The platform can also be used as an e-learning tool to support data mining courses and for defining interesting student projects. In this paper we discuss the architecture of Knowledge Pit and highlight its main functionalities. We also overview some of the already finished data challenges that were organized using our web platform.

Key words: data mining competitions, collaborative research, web platform, e-learning

1 Introduction

In this short paper we briefly describe a web platform, called Knowledge Pit, created in order to support organization of data mining competitions. On the one hand, this platform is appealing to members of the machine learning community for whom competitive challenges can be a source of new interesting research topics. Solving real-life complex problems can also be an attractive addition to academic courses for students who are interested in practical data mining. On the other hand, setting up a publicly available competition can be seen as a form of outsourcing the task to the community. This can be highly beneficial to the organizers who define the challenge, since it is an inexpensive way to solve the problem which they are investigating. Moreover, an open data mining competition can become a bridge between domain experts and data analysts. In a longer perspective, it may leverage a cooperation between industry and academic researchers.

* The authors are partially supported by Polish National Science Centre grant DEC-2012/05/B/ST6/03215 and by Polish National Centre for Research and Development grants PBS2/B9/20/2013 and O ROB/0010/03/001.
2 System Architecture

The Knowledge Pit platform is designed in a modular way, on top of an open-source e-learning platform Moodle.org [1] and as such, it follows the best practices of a software development. The current modules of the platform include user accounts management system, competition management subsystems, time and calendar functionalities, communications features (i.e. forums and messaging subsystems), and a flexible interface for connecting automated evaluation services prepared to assess contestants’ submissions.

Figure 1 shows an architecture schema of the Knowledge Pit platform. Its two main parts are the platform’s engine located at a dedicated server and the evaluation subsystems. Currently, Knowledge Pit is hosted on a server belonging to Polish Information Processing Society (http://pti.org.pl/) and is located in the fedcis.org domain.

The two main parts of the platform are the platform’s engine and the evaluation subsystems. The first one provides interfaces for defining and maintaining of data challenges, management of user’s profiles, submissions and private files, maintaining Leaderboards, and the internal messaging systems (competition forums, chats, as well as email and notification sending services). It is based on a very popular solution stack, i.e. Apache, MySQL and PHP [6, 9, 2]. Together they constitute a bridge between the platform and different groups of users (guests, participants of competitions, moderators and organizers of particular challenges, managers and administrators of the system).

The second part of the platform is responsible for assessment of solutions submitted by participants of particular competitions. Due to a flexible communication mechanism, this service may be distributed among several independent workstations, which guarantees the scalability of the evaluation process. Since evaluating submissions for

---

3 A competition’s Leaderboard is an on-line ranking of participants competing in that particular data challenge.
some competitions may require a lot of resources (e.g. memory, CPU time, disc I/O or database connections), this is a very important aspect of system’s architecture. For example, the assessment of a single submission to AAIA’14 Data Mining Competition required constructing several Naive Bayes classification models for a data table consisting of 50,000 objects and testing their performance on a different table with 50,000 objects described by 11,852 conditional attributes [3]. In that case, distribution of the required computations allowed for nearly real-time evaluation, even during the most busy moments of the competition.

Another advantage of separating the evaluation subsystems from the platform’s engine is that it may be implemented in any suitable programming language, as a script or a stand alone compiled application that can use any external libraries. In this way, the responsibility for preparation of a suitable evaluation procedure can be delegated to organizers of individual competitions. In such a case, the only requirement for the implementation of the evaluator is that it should maintain a correct protocol of information exchange with the platform’s engine. This flow of responsibilities frees Knowledge Pit from the things which it cannot cope with in a generic way. It also gives competition organizers a very flexible method of expressing their data mining task in a form of a fully customizable evaluation procedures. For instance, the evaluation procedure can be implemented in R language [8], in a form of a script that runs independently on several machines.

3 Examples of Data Challenges Hosted by Knowledge Pit

Knowledge Pit inaugurated in the beginning of 2014 and since then continues to organize successful data mining competitions in cooperation with international conferences. By June 2015 it had hosted 4 major competitions and a few local student projects. It currently has over 700 active users who participated in at least one data challenge and this number grows with every new competition. Below we list the recent competitions and shortly describe their scope. Typically, after completion of a contest, its overview and detailed descriptions of top solutions are published in proceedings of the associated conference.

3.1 AAIA’14 Data Mining Competition

AAIA’14 Data Mining Competition: Key risk factors for Polish State Fire Service took place between February 3, 2014 and May 7, 2014. In this challenge the focus was on the feature selection problem and the data came from the public safety domain. We asked members of the machine learning community to identify characteristics extracted from the EWID reports [5], which are useful for predicting whether any people were harmed during a given incident.

4 Web page: https://knowledgepit.fedcsis.org/contest/view.php?id=83
3.2 AAIA’15 Data Mining Competition

AAIA’15 Data Mining Competition: Tagging Firefighter Activities at a Fire Scene\(^5\) took place between March 9, 2015 and June 5. It was a continuation of the contest initiated during the previous edition of the data challenge associated with International Symposium on Advances in Artificial Intelligence and Applications (the AAIA conference series) [3]. The topic was related to real-time screening of firefighters’ vital functions and monitoring of ongoing physical activities at the incident scene [7].

3.3 PAKDD’15 Data Mining Competition

PAKDD’15 Data Mining Competition: Gender Prediction Based on E-commerce Data\(^6\) took place between March 23, 2015 and May 3 of the same year. The task in this competition was to reconstruct the information about user’s gender from product viewing logs from an on-line store. The data set was obtained from simulations of product viewing activities for user with known gender and was provided by FTP Group - the leading information and communication technology enterprise in Vietnam. The results of this competition were presented at a major Asia-Pacific data mining conference PAKDD’15 and were acclaimed by the industry representatives from FTP Group.

3.4 IJCRS’15 Data Challenge

IJCRS’15 Data Challenge: Mining Data from Coal Mines\(^7\) started April 13, 2015 and lasted until June 25, 2015. The task was to come up with a prediction model which could be effectively applied to foresee warning levels of methane concentrations at three methane meters placed in a longwall of the mine [4]. The data used in the competition came from an active Polish coal mine. They consisted of multivariate time series corresponding to readings of sensors used for monitoring the safety conditions at the longwall.

4 Future Development of the Platform

In this paper we briefly described our data challenge platform called Knowledge Pit. It is worth noticing that this non-commercial project is far from complete. We are continuously searching for new topics of data mining contests related to important practical issues. We are also working on developing new features and functionalities for our platform. One example of such a feature is a support for an evaluation system that not only assesses submissions of participants with regard to their predictive quality but also tries to grasp adaptiveness of the proposed solutions, i.e. how fast they can produce results with sufficient quality and how much training data do they need.

\(^5\) Web page: https://knowledgepit.fedcsis.org/contest/view.php?id=106
\(^6\) Web page: https://knowledgepit.fedcsis.org/contest/view.php?id=107
\(^7\) Web page: https://knowledgepit.fedcsis.org/contest/view.php?id=109
References

Toward Synchronization of EEG and Eye-tracking Data
Using an Expert System
Extended Abstract

Boleslaw Jaskula\textsuperscript{1}, Krzysztof Pancerz\textsuperscript{1,2}, and Jaroslaw Szkola\textsuperscript{1}

\textsuperscript{1} University of Information Technology and Management
Sucharskiego Str. 2, 35-225 Rzeszow, Poland
{bjaskula,jszkola}@wsiz.rzeszow.pl
\textsuperscript{2} University of Management and Administration
Akademicka Str. 4, 22-400 Zamosc, Poland
kpancerz@wszia.edu.pl

Abstract. In the paper, we present a new approach to solving the problem of combining data coming from eye-tracking and electroencephalography (EEG). This is a challenging problem in neurocognitive research. An important issue added to the process of combining data received from both devices is their synchronization. The created software system solving the problem is based on a specialized expert system.

Key words: electroencephalography, EEG, eye-tracking, expert system, data synchronization

1 Synchronization of EEG and Eye-tracking Data

Combining eye-tracking and electroencephalography (EEG) is one of the most fascinating and challenging problems. Scientists are dealing with this problem in order to close the gap between the psychological mental processes leading to the sensations we experience in everyday life and their underlying physiological biochemical processes. They are trying to understand perception, the process by which our brain makes sense out of the signals coming from our senses. Eye-tracking alone is not sufficient because it delivers to us only psychophysical data. It provides valuable information about the gaze location, but does not provide any information about neuronal activity. On the other hand, EEG, measuring neuronal activity in humans, does not directly provide information about the gaze position. For this reason, projects combining EEG and eye-tracking into one common setup have been started by scientists (see [6]).

Electroencephalography (EEG) is a well-established non-invasive technique for brain monitoring with high temporal resolution and relatively low cost. As such, EEG has proven to be a critical monitoring and diagnostic tool in the clinic [11]. EEG is also a popular research tool among scientists for evaluating somatosensory responses to stimuli, error detection [4], and sleep or fatigue monitoring [3], [12], among other uses. Various EEG components in the temporal domain have been used to define distinct phases of cortical processing in response to stimulus presentation. Eye-tracking
is the technology offering the possibility of capturing visual behavior in real-time and monitoring locations of fixations within images [7]. Recently, eye tracking technology has become more accurate and user friendly. It has extended to various areas that led to a wide range of applications, cf. [5], [8].

There are various areas in which combining eye-tracking and EEG is used. First of all, we can distinguish the following ones:

- visual search which involves finding a target in the midst of distractors (see, for example, [9], [13]),
- consumer neuroscience which is an emerging interdisciplinary field that combines psychology, neuroscience, and economics to study how the brain is physiologically affected by advertising and marketing strategies (see, for example, [1], [2]),
- data collection [10].

There is a need to integrate two methods of gathering data, mentioned earlier, for neurocognitive research. The main problem arising in combining eye-tracking and EEG data is their synchronization. Until now, there is a lack of information in the literature about the fulfilling solution of the problem. In our approach, we propose to use a specialized expert system.

For the research purposes, we have developed a software system whose task is to synchronize data received from EyeTracker Tobii T60 and Emotiv EPOC EEG (Portable version 1.0). The main problem arising in this task is that both devices work independently and we can only capture the data separately and next save them to files. In case of Tobii T60, the software is a black-box and we have access only to the recorded data. Therefore, these data can be analyzed in further stages. In case of EPOC EEG, the API (Application Programming Interface) is available. It allows us to prepare a dedicated software system that uses the signals obtained from the sensors.

The following assumptions have been made in order to fulfil requirements for the dedicated software system:

1. The measurement data must be synchronized in time.
2. The devices can be connected to two different machines at the same time (recommended).
3. Measurements should have a high time resolution.
4. An expert system analyzes the received data searching for the selected features (correlations, patterns, etc.).

To achieve these goals, we have proposed the following solution. Each device must have a system clock synchronized with the selected time standard. In our research, we have used a solution based on time synchronization with the atomic clock. The freeware Atomic Clock Sync v.3.5 has been used. One can also run a local time server to synchronize the two devices. If we run two applications on a single machine, synchronization is not necessary. However, our experience shows that this is a data-intensive computing job and there are situations when records are not synchronized. Another advantage of execution of tasks on different machines is the possibility for running applications under different operating systems. As we have access to the API of EPOC EEG, we are able to get time resolution between the recorded samples with the accuracy no less than
0.1 also with respect to data received from Tobii T60. An expert system analyzes the synchronized data received from two different devices, searching for selected features of the signal. An additional advantage of this approach is the ability to export both raw and processed data in the CSV format, which allows us to perform analyses using other specialized tools.

Gathering synchronized data is necessary but not sufficient for performing the reasoning process. An important problem, with respect to the character of the measured signals, is their quality as well as noise level. In case of the eye-tracker signal, the precision is high and the measurement is stable. Meanwhile, in case of the EEG signal, data are noised and they can include signals from the surroundings. The level of the EEG signal depends on the region in the brain.

The expert system consists of three main parts:

– a signal filtering part,
– an activity pattern recognition part,
– a reasoning part based on the first order logic.

An important novelty of the proposed approach is the application of information on the optic nerve gathered from the eye-tracker for removing artefacts. This effect is difficult to obtain using only the EEG measurement.

References

Data Integration through Clustering and Finding Statistical Relations - Validation of Approach

Marek Jaszuk, Teresa Mroczek, and Barbara Fryc

University of Information Technology and Management, ul. Sucharskiego 2, Rzeszow, Poland, {mjaszuk,tmroczek,bfryc}@wsiz.rzeszow.pl

Abstract. The paper analyzes an approach to data integration based on finding statistical relations between data. The data used for experimenting comes from surveys collected from student groups. The practical problem that underlies this research is discovering the model of knowledge about students, which would allow for making predictions about their future educational success or failure. The obstacle is that data collected for different groups over years has different format and this makes it difficult to reuse the previously collected data. Thus we had to find a way, to overcome this difficulty and integrate the heterogeneous data. The paper analyzes the feasibility of integrating data using this method. Although based on particular application, the model of computations presented in the paper is of more general nature, and should be applicable in many other domains.

Key words: data integration, clustering, semantic class, correlation, survey

1 Introduction

Increase in popularity of information technologies brought us a large number of independently created and managed information systems. Such systems can contain similar information but coming from disparate sources which leads to information heterogeneity. This prevents from interoperability of such systems and their integration. Thus it is highly demanded to overcome the heterogeneities through some data integration technique, and data integration is one of the central problems in information systems.

Usually information heterogeneity can be considered on three levels: syntactic, structural and semantic. Our focus in this paper is on semantic heterogeneity and data integration on this level. The problem of semantic heterogeneity has been studied intensively in the past years (see e.g. [2, 3, 5, 6, 9]). Automatic identification of semantic relations between different data sets has been investigated [2] together with representation and using of identified relations for transferring data and query answering [1, 5, 9]. A prominent part of research is devoted to investigating the role of ontologies, which represent formally the conceptual structure of a given application domain. The ontologies are used for identifying and using semantic relations, necessary for representing information systems to be integrated. In this regard, an ontology works as an intermediary between heterogeneous data sources.

The main difficulty with using ontologies is that they are usually handcrafted by domain experts accompanied by ontology engineers. Any modification in an ontology requires human effort, and thus is inefficient. Another problem is that an ontology for
a given domain can be developed in many different ways, and its optimal structure depends on particular application. Thus a particular ontology is not always suitable for a specific problem to be solved. In consequence, a method which automatizes integration of data coming from different sources, is highly desired.

The application example discussed in this paper is integration of different versions of survey data. Solving this problem using standard ontology based techniques would require developing ontologies for the survey questions, and then matching the ontologies to integrate the data. The semantic space for such an ontology is huge, because changing even a single word in a survey question can change the interpretation of this question, and thus shifts the meaning of this question. Thus doing the task of data integration manually can be a challenge. We propose an alternative approach based on finding statistical relations between data. This technique has been applied to surveys made on students, but it is more general, and could be applied to many other kinds of data with similar structure.

The paper is organized as follows. Sec. 2 discusses the practical problem to be solved. In Sec. 3 we discuss, how the survey questions are formulated. In Sec. 4 the basic assumptions about statistical data representation and the definition of semantic space. Then, in Sec. 5, we demonstrate the correlations between two data sets obtained independently, in order to confirm validity of the presented approach. Finally in Sec. 6 we discuss the way of integrating data coming from different surveys using clustering techniques.

2 The Problem Formulation

The practical problem that underlies considerations presented in this paper, is integration of data coming from groups of students. The University of Information Technology and Management in Rzeszów (UITM), where we conduct the research, collects some basic data about the students in the computer system, like the date of birth, the gender, the grades, etc. However, these data are not sufficient when one wants to perform more sophisticated kind of reasoning about the students. In our case, the main interest are in the future study results (educational success) of the students, that are beginning their education. Such information is interesting both for the group as a whole, as well as for selected individuals. The potential value of such information is both for the university authorities, as well, as for the teaching stuff, because it allows for early identification of potential problems, or outstanding individuals, who require special treatment. One of the factors that indicate the potential success are the results in the preceding stage of education, like the secondary school. This is, however, not a complete information, and the educational success is influenced by many other factors. We assume that the missing information about the factors influencing educational success can be collected by carrying out surveys, with questions related to socio-economic situation of students, as well as their motivations and reasons to study. The problem is in itself interesting from the social sciences perspective, but this aspect will not be discussed here.

There is potentially a large number of details, that could be asked in such a survey. The problem is thus selection of questions to be included in the survey. Unfortunately, the survey cannot be to long, otherwise, the students would not be willing to fill it.
Thus the choice of the questions should be very careful. The most adequate collection of questions can be obtained through a trial and error method. Each survey has to be followed by statistical analysis, which would indicate the questions well correlated with the educational success categories. After several iterations we should be able collect the questions delivering desirable information. But even if we identify the questions, there is always a possibility, that in the future someone would like to incorporate some new questions in the survey. The reason for such a modification would be changes in the external situation, and identification of possible new factors, that could be relevant. All the well known reasoning methods are based on unified data sets, i.e. to make predictions for a new data set, this set should be composed of data in the same format, on which the reasoning machine has been trained. Any modification in the data, like introducing new questions, requires retraining of the reasoning machine on the new data. At the same time, all the previous data which are incompatible with the new format become useless. This is an important problem, because in this way we lose a huge amount of unique data, which were difficult to collect.

If one wants to avoid loosing the previously recorded data, it is necessary to match the data formats in some way. This is usually done through manual effort. One of the standard approaches to the problem is based on using semantic models (ontologies), which allow for integrating both the data sets through ontology mapping [9]. The difficulties related to using ontologies, have already been mentioned, and we want to avoid the direct ontology creation and mapping. The approach demonstrated in this paper tries to complete the task data integration by generating a set of classes for data automatically. The foundations for this method were described in [4].

3 The Surveys

The questions for the surveys were prepared by social sciences experts according to their best knowledge. They were not tailored specifically for our experiments, but just to collect the data about the students, like this is done for other kinds of investigation. The first survey, that we analyzed, contained 21 questions with different structure depending on the question specifics. The structure of the questions was organized to make them clear and understandable. Fig. 1 shows the first question, which is a choice between 10 values from the range 1-10, reflecting the opinion of the questioned person. Some of the questions were actually groups of the questions related to the same subject (Fig. 2). Thus the actual number of questions is much larger than 21, due to subquestions. Yet another type of questions are multiple choice questions (Fig. 3).
2. Why have you chosen to study at the University of Information Technology in Rzeszow? (where 1 means the least important, 10 the most important reason)

<table>
<thead>
<tr>
<th>Reason</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>the possibility of finding a job after finishing this university</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>an interesting course</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>an interesting specialization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the possibility of obtaining a scholarship</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>student’s activity, the ability of developing individual interests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reasonable tuition height</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reasonable distance from the place where I live</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>internationality of the university (the possibility to travel, foreign students)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the level of lecturers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the position of the university in rankings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ease of study</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>opinions of parents</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>opinions of friends</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I did not get to my dream university</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>other (what?)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2. The question which is a combination of multiple subquestions

4 The Data Model

4.1 The Survey Representation

As we can see, the questions are of different form, and have to be reduced into a homogeneous format to allow for treating them in the same way. We do this by separating each possible outcome of the questions (an answer), and treating it as a separate attribute. In this way, for the first question in Fig. 1, we get 10 different question/answer pairs i.e:

- 1. Are you satisfied with the current study - 1
- 2. Are you satisfied with the current study - 2
- ...

5. What factors would have to appear, to make you resign from the study? (choose not more than 3 factors)

<table>
<thead>
<tr>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>my bad financial situation</td>
</tr>
<tr>
<td>my bad grades</td>
</tr>
<tr>
<td>if another university offered better financial conditions</td>
</tr>
<tr>
<td>if another university offered better teaching level</td>
</tr>
<tr>
<td>if another university had lesser expectations from students</td>
</tr>
<tr>
<td>if my friends moved to another university</td>
</tr>
<tr>
<td>if there would be an opportunity to move to another country</td>
</tr>
<tr>
<td>if another university was closer to the place where I live</td>
</tr>
<tr>
<td>another reason, what?</td>
</tr>
</tbody>
</table>

Fig. 3. The multiple choice question
20. Are you satisfied with the current study? - 10

The rest of the questions are decomposed in the same way. In consequence the initial number of 21 questions transforms into the space of more than 500 question/answer pairs. The survey after completing by a student is a binary vector:

\[ O_i = \{ I^i_k : k \in 1, \ldots, M \}, \]

where \( O_i \) is the vector representing \( i \)-th surveyed student, \( I^i_k \) is the \( k \)-th coordinate of \( i \)-th survey vector. \( M \) is the number of possible question/answer pairs in the survey. The vector contains 1-s in positions representing the answers selected by a student, and 0-s for answers which were not selected.

### 4.2 Educational Success Categories

Except the surveys, we collected the data about results of education for each of the students that filled the survey. These data are available in the university computer system. The results of education are the grades, or the information that the study has been broken for some reason. These data is available not earlier, than after the end of the first semester, while the survey was completed in the beginning of the academic year. The survey data are collected in the first semester of the university course. In this way we are able to follow the results of the students from the beginning till the end of the study, and confront them with the survey answers.

Our method requires dividing the investigated students into a number of groups related to their educational success. We do this by applying hierarchic clustering \([7]\) within the space of grades that the students got. The applied method of clustering was chosen because it allows for generating different numbers of clusters, depending on the choice of the cut point on the hierarchy. We performed validation of a number of well known clustering algorithms, and most of them revealed comparable clustering quality. So this factor was not crucial for the choice of clustering method.

We had no a priori assumption about the number of clusters, so we decided to choose the cut point that generated 5 clusters. This number seemed suitable to our experiments, although we do not exclude the possibility of experimenting also with other numbers of clusters. There was also a number of students, who filled the survey, but had no grades, because their study had been broken. This class of students are of particular interest, because they represent the educational failure. In consequence we got 6 categories of students - 5 coming from clustering, and one of those, who resigned from study.

### 4.3 The Semantic Distance

Our purpose is integration of the survey data. To be able to integrate them automatically, we have to start from determining the semantic distance between survey answers. The context data, which allow for determining the distance, are the educational success

---

1 Implemented in the R Project [8]
categories. The measure of the distance is based on statistical distribution of the survey answers with respect to the context:

\[ P_{I_k} = (P_{k1}, P_{k2}, \ldots, P_{kN}), \quad (2) \]

where \( P_{I_k} \) is the frequency distribution vector of the survey answer \( I_k \) with respect to the \( N \) success categories (in our case there are 6 categories). \( P_{kn} \) is the frequency with which the answer numbered \( k \) \((I_k)\) was found in the success category numbered \( n \), \((n = 1, \ldots, N)\), i.e. the chance that student belonging to success category \( n \), chooses the answer \( k \).

The space spanned by the distributions \( P_{I_k} \) plays the role of semantic space. The direction of the distribution vector \( (2) \) represents the meaning of every survey answer. The semantic distance between two answers is measured by the angle between respective distribution vectors. For practical reasons it is more convenient to use the cosine of the angle between the vectors. The cosine is the semantic similarity measure which ranges between 0 and 1. This range results from the frequencies, which are non-negative values, and thus the angle between vectors never exceeds \( \pi/2 \). The answers with identical meaning have the maximal similarity equal to 1, and the answers with completely different meaning have similarity equal to 0. The semantic similarity \( S_{kl} \) between two answers \( I_k \) and \( I_l \) is calculated as:

\[ S_{kl} = \cos \alpha_{kl} = \frac{P_{I_k} \cdot P_{I_l}}{\|P_{I_k}\| \|P_{I_l}\|}, \quad (3) \]

where \( \alpha_{kl} \) is the angle between \( P_{I_k} \) and \( P_{I_l} \) vectors.

The justification for the thesis, that the semantic distance can be measured using \( (3) \) is the observation, that if there would be two questions in the survey, with identical meaning, they have to generate the same probability distribution (similarity=1). Otherwise that would mean that surveyed persons interpreted the questions differently, and thus their meaning is different. The other possibility is that the survey was filled randomly, but we believe that this is not the case. It should be also noted that the similarity equal to 1 does not always mean that the human interpretation of the questions is identical. It is possible, that the interpretation is different, but still generates similar distribution. In terms of semantic model, this can be interpreted as synonymic question and answer. No matter which of the synonyms is used in the survey, the result is the same. So for computational purposes the synonymic questions are not a problem, because the reasoning based on them will be the same. A more detailed discussion of motivations for using the space of probabilities as the semantic space can be found in [4].

5 Correlations of Probability Distributions

The survey data are collected to build a computational model based on the groups of students that are currently studying, in order to be able to make predictions about the students recruited in the future. This approach makes sense only when the statistical distributions are stable for student groups from subsequent years. Thus it is necessary to verify the stability of distributions.
To assess the stability of the frequency distribution (2) for every answer in the survey, we conducted the survey for two student groups in subsequent years. The two surveys were not identical. They contained a number of questions that were identical, and a number of questions that were different. For the purpose of verification, only the common part of the questions is useful, so the rest of the question/answer pairs is not used for the stability verification purpose. Computing distributions (2) requires information about the grades of the surveyed students, to classify each of them into one of the 6 previously assumed categories of educational success. We clustered the first (older) of the investigated groups, and used the the same clustering for the second (younger) group. In this way the have a consistent classification system for students from both of the groups.

Given the classification we can compute the distributions (2) for each of the two groups. Then we can compare the distributions by computing the cosine of the angle between the distribution vectors for the same answers (semantic distance), but obtained from two different groups:

\[
S_{kk}^{12} = \cos \alpha_{kk} = \frac{P_{1k} \cdot P_{2k}}{\|P_{1k}\| \|P_{2k}\|},
\]

where \(P_{1k}, P_{2k}\) are the frequency distributions obtained for group 1 and group 2 of the students respectively.

Ideally, the similarity (4) should be close to 1 for each of the answers. In reality this value spreads over the whole range of possible values (Fig. 4(a)). There is an observable number of answers, which correlate between the groups (semantic similarity close to 1). But there are also answers, which do not correlate at all.

We were able to identify the reason for low correlations easily - the answers with low correlation come from the answers rarely chosen by the students. Some of the answers were chosen by just a few persons, or even by no one. With such small numbers no reliable statistical distribution can be determined. Luckily one of the significant reasons for low numbers was easy to eliminate. This was the wide range of possible answers to particular questions - in many cases this range was set to 10 single choice values, like in questions in Figs. 1 and 2. The total number of students in each of the investigated groups was about 200, so statistically the average number of persons that should choose each of the answers should be 20. However, the students clearly preferred some answers than the others. Actually we anticipated this situation, and chosen such a wide range deliberately, because it is easy to reduce the range afterwards, in case if the original range did not work. So the range was reduced from 10 down to 5 possible choices. This reduced the number of the considered answers form 401 to 255, and immediately increased the number of students who chosen each of the answers. After computing the similarity between groups we got the similarity distribution presented in Fig. 4(b).

As it can be observed, the degree of highly correlated answers increased after reducing the range of answers (e.g. the number of answers with correlation higher than 0.9 increased from about 25% to about 50%). However, there are still answers which do not correlate well. Despite applying the answer reduction trick, there are still answers, which are less likely to be chosen by the students. According to our findings, this is the main reason for low correlations, which result from less statistically reliable distribu-
tions. There was still some space for reducing the range of answers, e.g. to 3 or even 2 possible choices, which can increase the level of correlations. Thus we applied further reduction of the range of answers to 3 possible choices (Fig. 4(c)). The result confirmed our suspicions - the correlations further increased, and the number of uncorrelated answers vanished completely. The lowest similarity between the answers was on the level of 0.44.

Although the results look much better than the initial, there is still some space for improvements, which is related to several issues. The first of them are the survey questions. Not all them have the kind of structure which allow for their easy reduction. Thus there are still question/answer pairs, which are unlikely to be chosen by the students. The possible ways of increasing the correlations, would be:

1. eliminate the weakly correlated answers - the risk is, that in this way we will eliminate valuable information referring to a relatively small number of students,
2. reformulate the questions in order to force the students, to choose some answers more frequently - this is something that we consider to do in the future years,
3. increase the number of investigated students - we investigated students only from the Information technology specialization. Due to limited number of students, increasing the number would require extending the research onto other specializations, which is possible, but we are not sure if students, from very different specializations, will generate the same probability distributions. This is an interesting topic for the future research,

4. decrease the number of student success groups - the division into 6 groups might be too fine grained, thus we consider decreasing it to e.g. 4 groups, which immediately increases the number of students in each of the groups, and makes the distribution more reliable.

To summarize the results of correlation investigation, we can state that if the number of students, who selected a particular answer is sufficiently large, then the frequency distribution with respect to the educational success categories, in a vast number of cases remains stable. Thus such an answer can be used as a reliable indicator of the possible success category.

6 Data Integration through Clustering

As already mentioned, a survey can potentially contain many different questions, and along time some questions could be replaced by others. This makes it difficult to reuse the knowledge collected in the previous years, because the evolution of questions could lead to a potentially large set of questions. Finding relations among such questions along the timeline is a difficult task. Thus we develop the mechanism, which should allow for integrating the the old versions of surveys, with the newly created ones. The basis for this task is the already introduced semantic space of frequency distributions (2) together with the similarity measure (3).

The basic semantic relation, that can be discovered, among survey answers is the synonymy relation, i.e. finding answers with the same meaning. This task can be completed with the clustering technique. We applied the hierarchic clustering again due to its flexibility, and possibility of selecting various levels of clustering granularity. We used the cosine distance, to measure the distance between clustered objects (the survey answers), because this is the assumed semantic similarity measure. Here again it is interesting to assess, whether the closely related questions indeed fall in the same category.

There is a number of testing scenarios that can be proposed here. Because we want to integrate the data coming from surveys obtained from subsequent years, the best approach is to check, whether the answers belonging to some cluster for one year, belong to the same cluster in the subsequent year. To verify this, we clustered the answers for the first of the surveyed groups. Then we calculated the frequency distribution (2) for each of the answers in the second of the surveyed groups. This allows for determining the cluster (obtained on the first group), that each of the answers collected in the second group belongs to.

In the ideal situation, all of the answers for the second group should belong to the same clusters as for the first group. The results revealed that the situation is more complex to analyze. First of all, there are huge differences in the number of objects in each
of the clusters. That of course depends on the cut level in the hierarchy. But in general, the majority of answers are grouped in several huge clusters. This is illustrated in Fig. 5 for the cut level in the clustering hierarchy equal to 0.1 (the parameter ranges between 0 and 1 - the lower the value, the larger number of clusters). The largest cluster contained 54 answers. The second group are the middle sized clusters (2 to 5), where the number of answers ranged between 31 and 12. The third group are the smallest clusters (6 to 14), where the number of answers ranged between 7 and 1. This is an interesting result, because it brings us insight into the nature of the gathered information. We can see, that the answers grouped in the huge clusters, do not bring much new information. In fact, we could resign from using all the question/answer pairs belonging to such clusters, and leave just one of them for each of the clusters. This would reduce the survey complexity significantly. More interesting are the answers grouped in small clusters. Their uniqueness indicate, that they bring some valuable information about the students, which distinguishes them from the others. This also indicates the possible regions, in which the survey could be extended to gather more useful information.

![Fig. 5. The distribution of the number of answers in particular clusters](image)

What refers to the basic question, which is the membership of the same questions to the same clusters, we found that indeed, a huge number of questions belong to the same clusters. It is no surprise, that the key factor that influences that, is the cosine distance between the answers, for different sets. The closer the answers between the groups of students are, the larger chance, that they belong to the same cluster. Uncorrelated answers are unlikely to belong to the same clusters. Thus providing conditions, in which the collected data are highly correlated for subsequent years, is the key factor to guarantee high reliability of the data model. The number of answers, that matches particular clusters, of course depends of the free parameter - the cut point in the hierarchic clus-
tering. The lower the cut point, the more detailed clustering, and the more mismatching
answers. Together with increasing the cut point, the number of matches rises.

7 Conclusions

The paper investigated the problem of data integration on the example of data coming
from student surveys. For this purpose we defined a semantic space, which allows for
computing the similarity between the survey answers. This concept allows for identifying
answers with close meaning, which is the first step to integrating the data.

The main focus of this paper was to verify if this approach is reliable, and could
be used for integrating this kind of data. The correlations obtained for two subsequent
years, for which the survey was conducted, indicate that indeed - the statistical distribu-
tions for particular questions exhibit high similarities. Thus the approach can be the
basis for data integration. Although there are still some question/answer pairs, which
do not correlate well. We indicated the ways of dealing with the situation to improve
the results.

The other open question is the clustering method to be used to group the answers.
In this paper we used the hierarchic clustering, because of its flexibility in steering the
granularity with the cut point of the clustering hierarchy. But also other methods should
be tested. This is especially important when we realize, that the radius of the clusters
could be an important factor influencing the size of particular clusters. Unfortunately in
hierarchic clustering we have no direct influence on the radius.

The presented methodology of data integration was demonstrated on a particular
application example, but its nature is universal. It can be applied to any kind of data,
where we have information about a group of entities, and the entities can be classified
into a number of categories. This is a very wide category of problems, so there is a lot
of work to do, to analyze the results delivered by our methodology.

Acknowledgments. This work is the result of a project titled “Intelligent meth-
ods of analyzing chances and threats in the educational process” (project no. UDA-
RPPK.01.03.00-18-019/12-00), co-financed by the European Union from the European
Regional Development Fund and from the Budget within the Regional Operational Pro-
gramme for the Podkarpackie Region for the years 2007-2013.

References

1. Bellahsene, Z., Bonifati, A., Rahm, E.: Schema matching and mapping. Springer-Verlag,
   Berlin Heidelberg (2011)
   Clustering Technique. In: 3-rd International Conference on Data Management Technologies
Selected Methods of Combining Classifiers, when Predictions Are Stored in Probability Vectors, in a Dispersed Decision-Making System

Malgorzata Przybyla-Kasperek

University of Silesia, Institute of Computer Science, Bedzinska 39, 41-200 Sosnowiec, Poland
malgorzata.przybyla-kasperek@us.edu.pl
http://www.us.edu.pl

Abstract. Issues that are related to decision making that is based on dispersed knowledge are discussed in the paper. A system, that was proposed in the article [12], is used in this paper. In the system the process of combining classifiers in coalitions is very important and negotiation is applied in the clustering process. The main aim of the article is to compare the results obtained using five different methods of conflict analysis in the system. All these methods are used if the individual classifiers generate probability vectors over decision classes. The most popular methods are considered: a sum, a product, a median, a maximum and a minimum rules. In the paper, tests, which were performed on data from the UCI repository, are presented. The best methods in a particular situation are indicated. It was found out that some methods do not generate satisfactory results when there are dummy agents in a dispersed data set. That is there are undecided agents who assign the same probability value to many different decision values.

Key words: decision support system, dispersed knowledge, conflict analysis, sum rule, product rule, median rule, maximum rule, minimum rule

1 Introduction

The effectiveness of decision making based on dispersed knowledge is an increasingly vital issue. When important economic, medical, enterprise management, business and risk assessment or political issues are to be resolved, groups rather than individuals are employed to make high-quality decisions. Compared to individual decision makers, groups have access to more and a broader range of information, which is distributed among group members. So making decisions based on dispersed knowledge is more difficult, demanding and requires new methods of inference. In this paper an approach proposed in the article [12] is used to make decisions on the basis of dispersed knowledge. The aim of the paper is to investigate the use of five selected conflict analysis methods in the system with dynamically generated clusters. The problem of conflict analysis arises because the inference is being conducted in groups of knowledge bases. Five methods known from the literature [3, 5, 6] were used to conflict analysis: the sum rule, the product rule, the median rule, the maximum rule and the minimum rule.
Several attempts have already been made to solve the problem of decision making that is based on dispersed knowledge, group decisions and negotiations. In the paper [1], a group decision-making (GDM) problem in which linguistic information is used is considered. The method, which is based on granular computing and pairwise comparisons, was used to consider GDM situations that were defined in heterogeneous contexts, that is, situations in which the experts have different backgrounds and levels of knowledge about the problem. A discussion of pairwise comparisons of objects was presented in the paper [4]. The concept of distributed decision-making is widely discussed in the papers [13, 14]. In addition, the problem of using distributed knowledge is discussed in many other papers [2, 15, 16]. In the paper [18], an approach was proposed in which many classifying agents are generated by using fast approximation heuristics, after which a classification system is constructed by selecting the optimal subset of agents. This paper describes a different approach to the global decision-making process. We assume that the set of local knowledge bases that contain information from one domain is pre-specified. The only condition which must be satisfied by the local knowledge bases is to have common decision attributes.

2 A Brief Overview of a Dispersed Decision-Making System

The concept of a dispersed decision-making system is being considered by the author for several years. In the first stage of studies the considerations were directed to a system with a static structure [10, 17]. In recent papers a system with a dynamic structure has been proposed [11, 12]. During the construction of this system’s structure a negotiation stage is used. The main assumptions, notations and definitions of the system are described below. A detailed discussion can be found in the paper [12].

We assume that the knowledge is available in a dispersed form, which means in a form of several decision tables. Each local knowledge base is managed by one agent, which is called a resource agent. We call $ag$ in $Ag = \{ag_1, \ldots, ag_n\}$ a resource agent if he has access to resources represented by a decision table $D_{ag} := (U_{ag}, A_{ag}, d_{ag})$, where $U_{ag}$ is a finite nonempty set called the universe; $A_{ag}$ is a finite nonempty set of conditional attributes, $V_{ag}^a$ is a set of attribute $a$ values; $d_{ag}$ is referred to as a decision attribute, $V_{ag}^d$ is called the value set of $d_{ag}$. We want to designate homogeneous groups of resource agents. The agents who agree on the classification for a test object into the decision classes will be combined in the group. It is realized in two steps. At first initial coalitions are created. Then the negotiation stage is implemented. These two steps are based on the test object classification carried out by the resource agents. For each agent the classification is represented as a vector of values, whose dimension is equal to the number of decision classes. This vector will be defined on the basis of certain relevant objects. That is the objects from the decision tables of agents that carry the greatest similarity to the test object. From decision table of resource agent $D_{ag}, ag \in Ag$ and from each decision class, the smallest set containing at least $m_1$ objects for which the values of conditional attributes bear the greatest similarity to the test object is chosen. The value of the parameter $m_1$ is selected experimentally. Then for each resource agent $i \in \{1, \ldots, n\}$ and the test object $x$, a $c$-dimensional vector $[\bar{\mu}_{i,1}(x), \ldots, \bar{\mu}_{i,c}(x)]$ is generated. The value $\bar{\mu}_{i,j}(x)$ is equal to
the average value of the similarity of the test object to the relevant objects of agent \(ag_i\), belonging to the decision class \(v_j\). In the experimental part of this paper the Gower similarity measure [11] was used. This measure enables the analysis of data sets that have qualitative, quantitative and binary attributes. On the basis of the vector of values defined above, a vector of the rank is specified. The vector of rank is defined as follows: rank 1 is assigned to the values of the decision attribute that are taken with the maximum level of certainty. Rank 2 is assigned to the next most certain decisions, etc. Proceding in this way for each resource agent \(ag_i, i \in \{1, \ldots, n\}\), the vector of rank is specified. The vector of rank is defined as

\[
\phi_{v_j}^{x}(ag_i, ag_k) = \begin{cases} 0 & \text{if } \tilde{r}_{i,j}(x) = \tilde{r}_{k,j}(x) \\ 1 & \text{if } \tilde{r}_{i,j}(x) \neq \tilde{r}_{k,j}(x) \end{cases}
\]

where \(ag_i, ag_k \in Ag\). We also define the intensity of conflict between agents using a function of the distance between agents.

We define the distance between agents \(\rho^x\) for the test object \(x\):

\[
\rho^x(ag_i, ag_k) = \frac{\sum_{v_j \in V^d} \phi_{v_j}^{x}(ag_i, ag_k)}{\text{card}(V^d)}, \quad \text{where } ag_i, ag_k \in Ag.
\]

**Definition 1.** Let \(p\) be a real number, which belongs to the interval \([0, 0.5]\). We say that agents \(ag_i, ag_k \in Ag\) are in a friendship relation due to the test object \(x\), which is written \(R^+(ag_i, ag_k)\), if and only if \(\rho^x(ag_i, ag_k) < 0.5 - p\). Agents \(ag_i, ag_k \in Ag\) are in a conflict relation due to the test object \(x\), which is written \(R^-(ag_i, ag_k)\), if and only if \(\rho^x(ag_i, ag_k) > 0.5 + p\). Agents \(ag_i, ag_k \in Ag\) are in a neutrality relation due to the test object \(x\), which is written \(R^0(ag_i, ag_k)\), if and only if \(0.5 - p \leq \rho^x(ag_i, ag_k) \leq 0.5 + p\).

By using the relations defined above we can create groups of resource agents, which are not in conflict relation. The initial cluster due to the classification of object \(x\) is the maximum, due to the inclusion relation, subset of resource agents \(X \subseteq Ag\) such that \(v_{ag_i, ag_k \in X} R^+(ag_i, ag_k)\). In the second stage of clustering, limitations imposed on compatibility of agents are relaxed. We assume that during the negotiation, agents put the greatest emphasis on compatibility of ranks assigned to the decisions with the highest ranks. We define the function \(\phi_C^x\) for the test object \(x\):

\[
\phi_C^x(ag_i, ag_j) = \frac{\sum_{v_j \in Sign_{i,j}} |\tilde{r}_{i,j}(x) - \tilde{r}_{j,j}(x)|}{\text{card}(Sign_{i,j})}, \quad \text{where } ag_i, ag_j \in Ag \text{ and } Sign_{i,j} \subseteq V^d
\]

is the set of significant decision values for the pair of agents \(ag_i, ag_j\). In the set \(Sign_{i,j}\) there are the values of the decision, which the agent \(ag_i\) or agent \(ag_j\) gave the highest rank. During the negotiation stage, the intensity of the conflict between the two groups of agents is determined by using the generalized distance. The generalized distance between agents for the test object \(x\) is denoted by \(\rho_G^x\):

\[
\rho_G^x : 2^Ag \times 2^Ag \rightarrow [0, \infty).
\]

The value of the generalized distance function for two sets of agents \(X\) and \(Y\) is equal to the average value of the function \(\phi_C^x\) for each pair of agents \(ag, ag'\) belonging to the set \(X \cup Y\). This value can be interpreted as the average difference of the ranks assigned to significant decisions within the combined group of agents consisting of the sets \(X\) and \(Y\). For each agent \(ag\) that has not been included to any initial clusters, the generalized distance value is determined for this agent and all initial clusters, with
which the agent \(ag\) is not in a conflict relation and for this agent and other agents without coalition, with which the agent \(ag\) is not in a conflict relation. Then the agent \(ag\) is included to all initial clusters, for which the generalized distance does not exceed a certain threshold, which is set by the system’s user. Also agents without coalition, for which the value of the generalized distance function does not exceed the threshold, are combined into a new cluster. After completion of the second stage of the process of clustering we get the final form of clusters. For each cluster, a superordinate agent is defined, which is called a synthesis agent, \(as_j\), where \(j\)- number of cluster. \(As_x\) is a finite set of synthesis agents defined for the clusters that are dynamically generated for test object \(x\). Next, an approximated method of the aggregation of decision tables have been used to generate decision tables for synthesis agents (see [10–12] for more details). Based on these aggregated decision tables global decisions are taken using the methods of conflict analysis.

3 Methods of Conflict Analysis

In this article, we use five different methods of conflict analysis: the sum rule, the product rule, the median rule, the maximum rule and the minimum rule. These are well known and commonly used methods in group decision-making problems. They are discussed by various authors [3, 5–7]. The methods, that are used in this article, are simple, have low computational complexity and are easy to implement. These methods require no training and all classifiers are treated equally. In some applications, it may be undesirable, because the methods do not take into account the differences in the individual classifier capabilities. But, as we know from the literature, it is somewhat surprising to see how well these simple aggregation rules compete with the more sophisticated ones. The novelty that is proposed in this article involves the use of these five methods in a dispersed decision-making system that was briefly described above. All these methods are used if the individual classifiers generate vectors of probabilities instead of unique class choices. Therefore at first, on the basis of each aggregated decision table a vector of probabilities is generated. A \(c\)-dimensional vector of values \([\mu_{j,1}(x), \ldots, \mu_{j,c}(x)]\) is generated for each \(j\)-th cluster, where \(c\) is the number of all of the decision classes. This vector will be defined on the basis of relevant objects. From each aggregated decision table and from each decision class, the smallest set containing at least \(m_2\) objects for which the values of conditional attributes bear the greatest similarity to the test object is chosen. The value of the parameter \(m_2\) is selected experimentally. The value \(\mu_{j,i}(x)\) is equal to the average value of the similarity of the test object to the relevant objects form \(j\)-th aggregated decision table, belonging to the decision class \(v_i\). In this way, for each cluster the vector of probabilities is generated.

In the paper [6], it was proposed that the classifier outputs can be organized in a decision profile (DP) as the matrix. This is very clear and transparent way of classifiers outputs presentation. The decision profile is a matrix with dimensions \(\text{card}(As_x) \times c\), where \(As_x\) is a finite set of synthesis agents defined for the test object \(x\) and \(c\) is the
The decision profile is defined as follows

\[ DP(x) = \begin{bmatrix}
\mu_{1,1}(x) & \cdots & \mu_{1,i}(x) & \cdots & \mu_{1,c}(x) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\mu_{\text{card}(A_{s_x}),1}(x) & \cdots & \mu_{\text{card}(A_{s_x}),i}(x) & \cdots & \mu_{\text{card}(A_{s_x}),c}(x)
\end{bmatrix} \]

The \( j \)-th row of the matrix saves the output of \( j \)-th synthesis agents and the \( i \)-th column of the matrix saves support from agents \( A_{s_x} \) for decision class \( i \).

**The sum rule**

The sum rule consists in the designation for each decision class the sum of the probability values assigned to this class by each cluster. The set of decisions taken by the dispersed system is the set of classes which have the maximum of these sums. Thus, the set of global decisions that are generated using the sum rule is defined as follows

\[ \hat{d}_{WSD_{Ag}}(x) = \arg\max_{i \in \{1, \ldots, c\}} \left\{ \sum_{j=1}^{\text{card}(A_{s_x})} \mu_{j,i}(x) \right\} \]

**The product rule**

In the product rule for each decision class the product of the probability values is determined. The set of decisions taken by the dispersed system is the set of classes which have the maximum of these products

\[ \hat{d}_{WSD_{Ag}}(x) = \arg\max_{i \in \{1, \ldots, c\}} \left\{ \prod_{j=1}^{\text{card}(A_{s_x})} \mu_{j,i}(x) \right\} \]

The approach, that is used in this paper, has a small modification. The product rule is very sensitive to the most pessimistic prediction result of the base classifier. The worst is the situation in which one of the classifiers generate, for several decision classes, probability equal to 0. This situation is called a veto mechanism - one classifier is decisive. To eliminate this drawback, a rule was adopted, that if the probability for the decision class is equal to 0, then the values of probabilities are multiplied by \( 10^{-3} \) instead of 0.

**The median rule**

In the median rule for each decision class the median value of the probability values is determined. The median can be found by arranging all the values from lowest value to highest value and picking the middle one. If there is an even number of values, the median is defined to be the mean of the two middle values. The set of decisions taken by the dispersed system is the set of classes which have the maximum of these medians

\[ \hat{d}_{WSD_{Ag}}(x) = \arg\max_{i \in \{1, \ldots, c\}} \left\{ \text{med}_{j \in \{1, \ldots, \text{card}(A_{s_x})\}} \mu_{j,i}(x) \right\} \]

**The maximum rule and the minimum rule**

The maximum rule and the minimum rule consist in the designation for each decision class the maximum or the minimum value of the probability values assigned to this class by each cluster. The set of decisions taken by the dispersed system is the set of classes which have the maximum of these values. Thus, the sets of global decisions that are generated using these methods are defined as follows: the maximum rule

\[ \hat{d}_{WSD_{Ag}}(x) = \arg\max_{i \in \{1, \ldots, c\}} \left\{ \max_{j \in \{1, \ldots, \text{card}(A_{s_x})\}} \mu_{j,i}(x) \right\} \]

and the minimum rule

\[ \hat{d}_{WSD_{Ag}}(x) = \arg\max_{i \in \{1, \ldots, c\}} \left\{ \min_{j \in \{1, \ldots, \text{card}(A_{s_x})\}} \mu_{j,i}(x) \right\} \]

**Example 1.** Consider a dispersed decision-making system in which there are three decision classes \( V^d = \{v_1, v_2, v_3\} \) and, for a given test object, the set of synthesis agents
consists of five agents. We assume that the decision profile is as follows:

$$DP(x) = \begin{bmatrix}
0.5 & 0.4 & 0.3 \\
0.2 & 0.3 & 0.1 \\
0.6 & 0.4 & 0.2 \\
0.4 & 0.5 & 0.4 \\
0.5 & 0.6 & 0.3
\end{bmatrix}$$

Applying each of the methods columnwise, we obtain as a result the vectors and the sets of global decisions. The sum rule: \([2.2, 2.2, 1.3]\), \(\hat{d}_{WSD^{dyn}}(x) = \{v_1, v_2\}\); the product rule: \([0.012, 0.014, 0.001]\), \(\hat{d}_{WSD^{dyn}}(x) = \{v_2\}\); the median rule: \([0.5, 0.4, 0.3]\), \(\hat{d}_{WSD^{dyn}}(x) = \{v_1\}\); the maximum rule: \([0.6, 0.6, 0.4]\), \(\hat{d}_{WSD^{dyn}}(x) = \{v_1, v_2\}\); the minimum rule: \([0.2, 0.3, 0.1]\), \(\hat{d}_{WSD^{dyn}}(x) = \{v_2\}\). As can be seen, these methods provide different results, and sometimes more than one decision value is generated. This ambiguity of the methods are discussed in more detail in the next section.

4 Results of Experiments

The aim of the experiments is to compare the quality of the classification made by the dispersed decision-making system using five the most popular methods of combining the prediction’s results when vectors of probabilities are generated by the base classifiers. The sum, the product, the median, the maximum and the minimum rules were considered. For the experiments the following data, which are in the UCI repository [19], were used: Soybean Data Set and Vehicle Silhouettes data set. In order to determine the efficiency of inference of the dispersed decision-making system with respect to the analyzed data, the Vehicle Silhouettes data set was divided into two disjoint subsets: a training set and a test set. The Soybean data set is available on the UCI repository website in a divided form: a training and a test set. A numerical summary of the data sets is as follows: Soybean: # The training set - 307; # The test set - 376; # Conditional - 35; # Decision - 19; Vehicle Silhouettes: # The training set - 592; # The test set - 254; # Conditional - 18; # Decision - 4. Because the available data sets are not in the dispersed form, in order to test the dispersed decision-making system the training set was divided into a set of decision tables. Divisions with a different number of decision tables were considered. For each of the data sets used, the decision-making system with five different versions (with 3, 5, 7, 9 and 11 decision tables) were considered. For these systems, we use the following designations: \(WSD^{dyn}_{Ag_1}\) - 3 decision tables; \(WSD^{dyn}_{Ag_2}\) - 5 decision tables; \(WSD^{dyn}_{Ag_3}\) - 7 decision tables; \(WSD^{dyn}_{Ag_4}\) - 9 decision tables; \(WSD^{dyn}_{Ag_5}\) - 11 decision tables. Note that the division of the data set was not made in order to improve the quality of the decisions taken by the decision-making system, but in order to store the knowledge in a dispersed form. We consider the situation, that is very common in life, in which data from one domain are collected by different units as separate knowledge bases. For each data set we have 5 versions of the dispersion, therefore it can be said that 10 different dispersed data set were used for experiments. During the experiments, it turned out that not all analyzed methods of
combining individual classifiers, generate unequivocal decision. Sometimes when making
global decisions ties occur. It was noted that this situation occurs particularly for the
Soybean data set, when we are using the maximum and the minimum rule. Because we
want to observe and analyze such cases, an additional method of ties resolving were not
applied. But the appropriate classification measures were applied, which are adequate
to this situation. The measures of determining the quality of the classification are: estimator of classification error $e$ in which an object is considered to be properly classified if the decision class used for the object belonged to the set of global decisions generated by the system; estimator of classification ambiguity error $e_{ONE}$ in which object is considered to be properly classified if only one, correct value of the decision was generated to this object; the average size of the global decisions sets $d_{WSD^{dyn}}$ generated for a test set. In the description of the results of experiments for clarity some designations for algorithms and parameters have been adopted: $m_1$ - parameter which determines the number of relevant objects that are selected from each decision class of the decision table and are then used in the process of cluster generation; $p$ - parameter which occurs in the definition of friendship, conflict and neutrality relations; $A(m)$ - the approximated method of the aggregation of decision tables; $C(m_2)$ - the method of conflict analysis (the sum rule, the product rule, the median rule, the maximum rule or the minimum rule), with parameter which determines the number of relevant objects that are used to generate the probability vectors. The process of parameters optimization was carried out as follows. A series of tests for different parameter values were performed: $m_1 \in \{1, 6, 11, 16, 20\}$, $m, m_2 \in \{1, \ldots, 10\}$ and $p \in \{0.05, 0.15\}$. Thus, for each of the ten considered dispersed systems, 1000 tests were conducted ($1000 = 5 \cdot 10 \cdot 10 \cdot 2$).

From all of the obtained results, one was selected that guaranteed a minimum value of estimator of classification error ($e$), while maintaining the smallest possible value of the average size of the global decisions sets ($d_{WSD^{dyn}}$). In tables presented below the best results, obtained for optimal values of the parameters, are given.

The results of the experiments with the Soybean data set are presented in Table 1. In the table the following information is given: the name of dispersed decision-making system (System); the selected, optimal parameter values (Parameters); the algorithm’s symbol (Algorithm); the three measures discussed earlier $e, e_{ONE}, d_{WSD^{dyn}}$; the time $t$ needed to analyze a test set expressed in minutes. As can be seen, for the Soybean data set, unequivocal decisions are generated by the sum and the product rules. Which means that the average size of the global decisions sets is equal to 1. In the case of the median rule depending on chosen parameters, we get a unequivocal decision or a set of decisions having on average 1.5 decisions. For the maximum and the minimum rules regardless of the selected values of the parameters we get quite large set of decisions, sometimes average size is close to the value 2. For other analyzed values of the parameters even larger average number was observed. This ambiguity causes that these two methods are not very useful in the case of the Soybean data set. On the basis of detailed analysis of vectors of probabilities generated by the individual classifiers, it was concluded that the reason of this situation is that for the Soybean data there is a lot of dummy agents. That is there are undecided agents who assign the same probability value to many different decision values. Figure 1 shows a graphical comparison of the estimator of classification error for different dispersed systems of the Soybean data set.
Table 1. Summary of experiments results with the Soybean data set

<table>
<thead>
<tr>
<th>System Parameters</th>
<th>Algorithm</th>
<th>$e$</th>
<th>$e_{ONE}$</th>
<th>$d_{WSDDyn4e}$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The sum rule</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>A(3) C(3)</td>
<td>0.088</td>
<td>0.088</td>
<td>1</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 11, p = 0.15$</td>
<td>A(2) C(5)</td>
<td>0.093</td>
<td>0.093</td>
<td>1</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 11, p = 0.15$</td>
<td>A(3) C(9)</td>
<td>0.096</td>
<td>0.096</td>
<td>1</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>A(3) C(1)</td>
<td>0.082</td>
<td>0.133</td>
<td>1.059</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 11, p = 0.15$</td>
<td>A(1) C(1)</td>
<td>0.122</td>
<td>0.226</td>
<td>1.136</td>
</tr>
<tr>
<td><strong>The product rule</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 20, p = 0.15$</td>
<td>A(3) C(3)</td>
<td>0.085</td>
<td>0.085</td>
<td>1</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 16, p = 0.05$</td>
<td>A(5) C(4)</td>
<td>0.104</td>
<td>0.104</td>
<td>1</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>A(1) C(2)</td>
<td>0.122</td>
<td>0.234</td>
<td>1.157</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>A(1) C(2)</td>
<td>0.093</td>
<td>0.144</td>
<td>1.056</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 16, p = 0.05$</td>
<td>A(1) C(1)</td>
<td>0.138</td>
<td>0.237</td>
<td>1.144</td>
</tr>
<tr>
<td><strong>The median rule</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(3) C(9)</td>
<td>0.109</td>
<td>0.109</td>
<td>1</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 6, p = 0.15$</td>
<td>A(1) C(1)</td>
<td>0.008</td>
<td>0.239</td>
<td>1.644</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(2) C(2)</td>
<td>0.189</td>
<td>0.202</td>
<td>1.016</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 11, p = 0.15$</td>
<td>A(1) C(1)</td>
<td>0.008</td>
<td>0.271</td>
<td>1.628</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 11, p = 0.15$</td>
<td>A(2) C(3)</td>
<td>0.117</td>
<td>0.160</td>
<td>1.069</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 11, p = 0.15$</td>
<td>A(1) C(2)</td>
<td>0.048</td>
<td>0.348</td>
<td>1.574</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(7) C(2)</td>
<td>0.250</td>
<td>0.279</td>
<td>1.045</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>A(2) C(1)</td>
<td>0.098</td>
<td>0.426</td>
<td>1.460</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>A(5) C(5)</td>
<td>0.348</td>
<td>0.418</td>
<td>1.112</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(2) C(1)</td>
<td>0.250</td>
<td>0.535</td>
<td>1.404</td>
</tr>
<tr>
<td><strong>The maximum rule</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(4) C(5)</td>
<td>0.316</td>
<td>0.367</td>
<td>1.133</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(2) C(5)</td>
<td>0.449</td>
<td>0.601</td>
<td>2.064</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(5) C(8)</td>
<td>0.465</td>
<td>0.569</td>
<td>1.503</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(4) C(5)</td>
<td>0.590</td>
<td>0.737</td>
<td>1.521</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(2) C(2)</td>
<td>0.859</td>
<td>0.952</td>
<td>2.285</td>
</tr>
<tr>
<td><strong>The minimum rule</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>A(4) C(4)</td>
<td>0.162</td>
<td>0.205</td>
<td>1.721</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 11, p = 0.15$</td>
<td>A(2) C(6)</td>
<td>0.176</td>
<td>0.245</td>
<td>1.846</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 6, p = 0.15$</td>
<td>A(2) C(6)</td>
<td>0.205</td>
<td>0.285</td>
<td>1.965</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>A(6) C(1)</td>
<td>0.096</td>
<td>0.229</td>
<td>1.846</td>
</tr>
<tr>
<td>WSD$_{Agk}$</td>
<td>$m_1 = 1, p = 0.15$</td>
<td>A(3) C(1)</td>
<td>0.178</td>
<td>0.301</td>
<td>1.263</td>
</tr>
</tbody>
</table>
The results presented on the graph are divided into two groups. Methods that generate a set of decisions with the average size approximately equal to 1: the sum, the product and the median rules $d = 1$, and the methods that generate larger set of global decisions: the median $d = 1.5$, the maximum and the minimum rules. Based on the results it can be concluded that among the methods that generate unambiguous decisions, the best method is the sum rule, in second place is the product and in third place is the median rule. When a system’s user allows small ambiguity in the process of decisions making it is best to use the median rule. However, it should be noted that this method does well only in the case of a smaller number of resource agents. For a large number of agents 9 and 11 it does not generate good results. The minimum, and especially the maximum rule do not achieve satisfactory efficiency of inference, despite of significantly increased the average size of the global decisions sets. The results of the experiments with the Vehicle data set are presented in Table 2. As can be seen, for the Vehicle Silhouettes data set, unequivocal decisions are generated by all analyzed methods of combining classifiers’ predictions. Which means that the average size of the global decisions sets is equal to 1. On the basis of detailed analysis of vectors of probabilities generated by the individual classifiers, it was concluded that the reason of this situation is that for the Vehicle data set there are no dummy agents. That is there are no undecided agents who assign the same probability value to many different decision values. Figure 2 shows a graphical comparison of the estimator of classification error for different dispersed systems of the Vehicle Silhouettes data set. Based on the presented results it can be concluded that the sum, the product and the median rules are significantly better than the maximum and the minimum rules. It is difficult to say which method is the best. For the systems with 3 and 5 resource agents ($WSD_{Ag1}^{dyn}$, $WSD_{Ag2}^{dyn}$) the best is the sum rule, in second place is the product and in third place is the median rule. But for the systems with 7 and 9 resource agents ($WSD_{Ag3}^{dyn}$, $WSD_{Ag4}^{dyn}$), the best is the median rule, then the product and the sum rule. For the system with 11 resource agents ($WSD_{Ag5}^{dyn}$) these three methods give the same result. Summarizing the results presented in tables 1 and 2 it can be said that the best results from the examined methods achieve the sum
Table 2. Summary of experiments results with the Vehicle Silhouettes data set

<table>
<thead>
<tr>
<th>System</th>
<th>Parameters</th>
<th>Algorytm</th>
<th>$e_{OKE}$</th>
<th>$d_{WSDDyn}$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$WSD_{A4h}^{dyn}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>$A(3)C(4)$</td>
<td>0.240</td>
<td>0.240</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>$A(4)C(9)$</td>
<td>0.291</td>
<td>0.291</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 11, p = 0.05$</td>
<td>$A(4)C(6)$</td>
<td>0.252</td>
<td>0.252</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>$A(4)C(5)$</td>
<td>0.311</td>
<td>0.311</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 11, p = 0.05$</td>
<td>$A(4)C(1)$</td>
<td>0.268</td>
<td>0.268</td>
<td>1</td>
</tr>
</tbody>
</table>

The product rule

<table>
<thead>
<tr>
<th>System</th>
<th>Parameters</th>
<th>Algorytm</th>
<th>$e_{OKE}$</th>
<th>$d_{WSDDyn}$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>$A(3)C(4)$</td>
<td>0.244</td>
<td>0.244</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>$A(8)C(13)$</td>
<td>0.295</td>
<td>0.295</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 11, p = 0.05$</td>
<td>$A(4)C(6)$</td>
<td>0.248</td>
<td>0.248</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 11, p = 0.05$</td>
<td>$A(5)C(8)$</td>
<td>0.303</td>
<td>0.303</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 11, p = 0.05$</td>
<td>$A(4)C(5)$</td>
<td>0.268</td>
<td>0.268</td>
<td>1</td>
</tr>
</tbody>
</table>

The median rule

<table>
<thead>
<tr>
<th>System</th>
<th>Parameters</th>
<th>Algorytm</th>
<th>$e_{OKE}$</th>
<th>$d_{WSDDyn}$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>$A(7)C(6)$</td>
<td>0.252</td>
<td>0.252</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>$A(10)C(1)$</td>
<td>0.303</td>
<td>0.303</td>
<td>1.004</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>$A(4)C(8)$</td>
<td>0.240</td>
<td>0.240</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 11, p = 0.05$</td>
<td>$A(4)C(3)$</td>
<td>0.283</td>
<td>0.283</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>$A(4)C(5)$</td>
<td>0.268</td>
<td>0.272</td>
<td>1.004</td>
</tr>
</tbody>
</table>

The maximum rule

<table>
<thead>
<tr>
<th>System</th>
<th>Parameters</th>
<th>Algorytm</th>
<th>$e_{OKE}$</th>
<th>$d_{WSDDyn}$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>$A(4)C(7)$</td>
<td>0.252</td>
<td>0.252</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>$A(5)C(4)$</td>
<td>0.327</td>
<td>0.327</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>$A(4)C(4)$</td>
<td>0.339</td>
<td>0.339</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>$A(5)C(5)$</td>
<td>0.362</td>
<td>0.366</td>
<td>1.004</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 11, p = 0.05$</td>
<td>$A(2)C(3)$</td>
<td>0.358</td>
<td>0.394</td>
<td>1.039</td>
</tr>
</tbody>
</table>

The minimum rule

<table>
<thead>
<tr>
<th>System</th>
<th>Parameters</th>
<th>Algorytm</th>
<th>$e_{OKE}$</th>
<th>$d_{WSDDyn}$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 6, p = 0.05$</td>
<td>$A(3)C(2)$</td>
<td>0.287</td>
<td>0.287</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>$A(2)C(7)$</td>
<td>0.382</td>
<td>0.382</td>
<td>1</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 11, p = 0.05$</td>
<td>$A(4)C(2)$</td>
<td>0.280</td>
<td>0.283</td>
<td>1.004</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 11, p = 0.05$</td>
<td>$A(1)C(1)$</td>
<td>0.378</td>
<td>0.382</td>
<td>1.008</td>
</tr>
<tr>
<td>$WSD_{A4h}^{A4h}$</td>
<td>$m_1 = 1, p = 0.05$</td>
<td>$A(5)C(1)$</td>
<td>0.343</td>
<td>0.366</td>
<td>1.024</td>
</tr>
</tbody>
</table>
and product rule. These methods produce unambiguous results with the best observed efficiency of inference. During the experiments, the methods of combining classifiers’ predictions without the use of a system with dynamically generated clusters were analyzed. The global decision-making process was as follows. On the basis of decision tables of resource agents the probability vectors were generated, and then one of the five discussed methods of combining predictions was used. The obtained results show that the use of a system with dynamically generated clusters significantly improves the efficiency of inference. However, due to the limited length of the article, results of these experiments are not presented here.

5 Conclusion

In this article, five different methods of conflict analysis were used in the dispersed decision-making system: the sum rule, the product rule, the median rule, the maximum rule and the minimum rule. In the experiments, which are presented, dispersed data have been used: Soybean data set and Vehicle Silhouettes data set. Based on the presented results of experiments it can be concluded that the sum and the product rules produce the best results from the methods that were examined. The maximum and the minimum rules produce the worst results. Especially, the results are not interesting, when dummy agents are present in a dispersed data set. It appears that the methods of conflict analysis should be applied in different situations and it seems to be possible to use more than one approach in the same session. Initially, several methods could be used simultaneously to generate the sets of global decisions, then these sets can be merged in some way. This could be an option for improving the overall accuracy. It is planned to investigate such an approach in a future work.

References

Outliers Elimination for Error Correction Algorithm Improvement

Janusz Kolbusz and Pawel Rozycki
University of Information Technology and Management in Rzeszow
jkolbusz@wsiz.rzeszow.pl, prozycki@wsiz.rzeszow.pl
http://wsiz.rzeszow.pl

Abstract. Neural networks are still very important part of artificial intelligence. RBF networks seems to be more powerfull than that based on sigmoid function. Error Correction is second order training algorithm dedicated for RBF networks. The paper proposes method for improvement this algorithm by elimination of inconsistent patterns. The approach is also experimentally confirmed.

Key words: Error Correction, ErrCor, outliers, RBF networks, training algorithms

1 Introduction

Our civilization encounters increasingly complex problems that often exceeds human capabilities. Until recently, the aim was to create artificial intelligence systems so perfect, like a man. Currently, we are able to create intelligent learning systems exceeding the intelligence of the people. For example, we can create a model and predict the behavior of complex natural processes, which cannot be described mathematically. We can also identify economic trends that are invisible to humans. In order to efficiently model complex multidimensional nonlinear systems should be used unconventional methods. For given multidimensionality and nonlinear nature, algorithmic or statistical methods give unsatisfactory solutions. Methods based on computational intelligence allow to more effectively address complex problems such as foreseeing of economic trends, modeling natural phenomena, etc. To a greater extent than now harness the power of this type of network, you must:

– understand the neural network architecture and its impact on the functioning of the system and the learning process.

– find effective learning algorithms that allow faster and more effectively teach a network using its properties.

Both of problems are strictly connected.

The commonly used network MLP (Multi-Layer Perception) have relatively limited capacity[1]. It turns out that the new neural networks such as BMLP (Bridged MLP)[1,2] or DNN (Dual Neural Networks) [2] with the same number of neurons area to solve problems 10 or 100 times more complex [2,3].
A way of connecting neurons in the network is fundamental. For example, if you combine 10 neurons in the most commonly used three-tiered architecture MLP (with one hidden layer) the biggest problem that can be solved with such network is the problem of Parity-9 type. If the same 10 neurons are connected in the FCC architecture (Fully Connected Cascade), it is possible to solve the problem of Parity-1023 type. As can be seen a departure from the commonly used MLP architecture, while maintaining the same number of neurons, increases network capacity, even a hundred times [2-4]. The problem is that the commonly known learning algorithms, such as EBP (Error Back Propagation) [5], or LM (Levenberg-Marquardt), are not able to effective train these new highly efficient architectures. It is important to note that not only architecture, but also the training algorithm is needed to solve given problem. Currently, the only algorithm that is able to teach the new architecture is the NBN (Neuron by Neuron) published recently in [6-8]. This algorithm can be used for all architectures with arbitrarily connected neurons, including BMLP and DNN. This algorithm works well solving the problems impossible to solve by other algorithms.

Already now we can build intelligent systems, such as artificial neural networks, setting weights with random values initially, and then use an algorithm that will teach this system adjusting these weights in order to solve complex problems. It is interesting that such a system can achieve a higher level of competence than teachers. Such systems can be very useful wherever decisions are taken, even if the man is not able to understand the details of their actions. Neural networks helped solve thousands of practical problems. Most scientists used the MLP and the EBP algorithm. However, since the EBP algorithm is not efficient, usually using inflated the number of neurons which meant that the network with a high degree of freedom to consume their capabilities to learn the noise. Consequently, after the step of learning system was score responsive to the patterns that are not used during the learning, and it resulted in frustration. A new breakthrough in intelligent systems is possible due to new, better architectures and better, more effective learning algorithms.

2 Training Algorithms

Currently, the most effective and commonly known ANN training algorithms are algorithms based on LM[8]. Unfortunately, the LM algorithm is not able to teach other architectures than MLP. Because the size of Jacobian, which must be processed as proportional to the number patterns of learning. It means that LM algorithm may be used only for relatively small problems. Our newly developed second-order learning algorithm NBN [6-8] is even slightly faster than LM and allows to solve problems with a virtually unlimited number of patterns, and it may very effectively teach new powerful architecture of ANN, such as BMLP, FCC, whether DNN [1]. Using the NBN we can solve much more complex problems with more powerful system architectures.

Training of RBF (Radial Basis Function) network with the second order algorithm is even more complicated than training sigmoidal networks where are needed only to adjusted weights. Our preliminary research shows that if we can teach widths and locations of RBF centers it is possible to solve many problems in just a few units of RBF instead of hundreds sigmoid neurons.
The discovery of the EBP algorithm [5, 9] started a rapid growth of computational intelligent systems. Thousands of practical problems have been solved with the help of neural networks. Although other neural networks are possible, the main accomplishments were noticed using feed forward neural networks using primarily MLP architectures. Although EBP was a real breakthrough, this is not only a very slow algorithm, but also it is not capable of training networks with super compact architectures [1, 6]. Many improvements to the EBP were proposed, but most of them did not address the main faults of EBP. The most noticeable progress was done with an adaptation of the LM algorithm to neural network training [3]. The LM algorithm is capable of training networks with 100 to 1000 fewer iterations. The above mentioned LM algorithm [3, 10] was adapted only for MLP architectures, and only relatively small problems can be solved with this algorithm because the size of the computed Jacobian is proportional to the number of training patterns multiplied by the number of network outputs. Several years ago adapted the LM algorithm to train arbitrarily connected feed forward.

ANN architectures [11], but still the problem of the number of patterns limitations in the LM algorithm remained unsolved until recently when we developed the NBN algorithm [5]. Now we have a tool which is not only very fast, but we can train using second order algorithm problems with basically an unlimited number of patterns. Also NBN algorithm can train compact close to optimal architectures which cannot be trained by the EBP algorithm.

Both technologies (SVM and ELM) are adjusting only parameters, which are easy to adjust, like output weights, while other essential parameters such as radiiuses of RBF units $\sigma_h$, and the location of centers of the RBF units $c_h$ are either fixed or selected randomly. As a consequence, the SVM and ELM algorithms are producing significantly more networks than needed. From this experiment one may notice that the SVR (Support Vector Regression) [12, 13] and the Incremental Extreme Learning Machine (I-ELM) [14], and the Convex I-ELM (CI-ELM) [15] need 30 to 100 more RBF units than the NBN [16], the ISO [17], and the ErrCor [18] algorithms. Another advantage of ErrCor is that there is no randomness in the learning process so only one learning process is needed, while in the case of SVM (or SVR) a lengthy and tedious trial and error process is needed before optimal training parameters are found.

3 Error Correction Algorithm Improvement

3.1 Error Correction Fundamentals

Error Correction (ErrCor) is the second order LM based algorithm that has been designed for RBF networks where as neurons RBF units with Gaussian activation function defined by (1) are used.

$$\phi_h(x_p) = \exp \left( - \frac{||x_p - c_h||^2}{\sigma_h} \right)$$

where: $c_h$ and $\sigma_h$ are the center and width of RBF unit $h$, respectively. $||\cdot||$ represents the computation of Euclidean Norm.
The output of such network is given by:

\[ O_p = \sum_{h=1}^{H} w_h \varphi_h(x_p) + w_o \]  \hspace{1cm} (2)

where: \( w_h \) presents the weight on the connection between RBF unit \( h \) and network output. \( w_o \) is the bias weight of output unit. Note that the RBF networks can be implemented using neurons with sigmoid activation function [19,20].

The main idea of the ErrCor algorithm is increasing the number of RBF units one by one and adjusting all RBF units in network by training after adding of each unit. New unit is initially set to compensate largest error in the current error surface and after that all units are trained changing both centers and widths as well as output weights. Details of algorithm can be found in [18]. As can be found in [18] [21] ErrCor algorithm had been successfully used to solve several problems like function approximation, classification or forecasting. The main disadvantage of ErrCor algorithm is long computation time caused mainly by requirement of training of whole network at each iteration.

3.2 ErrCor Improvement

Long computation time depends on many factors. One of the most important is number of patterns used in training. We can reduce their number by removing from training dataset outlier patterns that includes data inconsistent with rest of patterns. This approach has been used in [21] to eliminate patterns that contain unusual data like hurricanes, political or criminal events. Such operation allows not only to reduce number of patterns or time of training but also to improve training results achieving lower training error and better generalization. The important issue is how to identify inconsistent patterns (outliers). We suggest to remove patterns for which error has higher value than Outlier Threshold (OT) that can be arbitrary assumed value. In our experiments OT was current MERR (Mean Error) dependent value given by:

\[ OT = n \times MERR \]  \hspace{1cm} (3)

where \( n \) is typically in range (5-10).
Removing of outliers can be done after adding to network several number of units. Pseudo code of the enhanced ErrCor algorithm is shown below. Changes to original ErrCor algorithm [18] are bolded.

**Improved ErrCor pseudo code**

```plaintext
evaluate error of each pattern;
while 1
    C = pattern with biggest error;
    add a new RBF unit with center = C;
    train the whole network using ISO-based method;
    evaluate error of each pattern;
    calculate SSE = Sum of Squared Errors;
    if SSE < desired SSE
        break;
    end;

    after each N added RBF units remove outliers with error > OT;
end
```

Described mechanism has been successfully used in [21] to improve training process of RBF network for forecasting energy load. It allowed to achieve both better training error and validation error as well as lower training time.

![Fig. 2. Data for network learning - a) function Schwefel, b) noised function Schwefel](image)

## 4 Results of Experiments

To confirm suggested approach several experiments with different dataset and training parameters have been prepared. The first experiment was approximation of noised Schwefel function. Noised function has been built by adding random values to about
20% of randomly selected Shwefel function samples. Original and noised function is shown in Figure 2. In presented experiments number 503 of 2500 samples ware noised. Such created data has been divided into training and testing datasets in the ratio of 4 to 1, to give 2000 training and 500 testing patterns. First, the training process to has been prepared using original ErrCor algorithm and next repeated for different values of parameter OT (from 1.5 to 4.65) and parameter N (5 and 10). In all experiments number of RBF units have been limited to 30. Results contain training MSE (Mean Square Error) and testing MSE are shown in Table 1.

**Table 1. Results for approximation of Schwefel function with Improved ErrCor algorithm with different values of parameters N and OT**

<table>
<thead>
<tr>
<th>N</th>
<th>OT</th>
<th>Training MSE</th>
<th>Testing MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.5</td>
<td>0.000046</td>
<td>0.008399</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.000129</td>
<td>0.004438</td>
</tr>
<tr>
<td>5</td>
<td>2.5</td>
<td>0.000224</td>
<td>0.004411</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0.031460</td>
<td>0.034999</td>
</tr>
<tr>
<td>5</td>
<td>3.5</td>
<td>0.000182</td>
<td>0.004331</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>0.000619</td>
<td>0.004350</td>
</tr>
<tr>
<td>5</td>
<td>4.5</td>
<td>0.000605</td>
<td>0.004470</td>
</tr>
<tr>
<td>5</td>
<td>4.6</td>
<td>0.000654</td>
<td>0.004473</td>
</tr>
<tr>
<td>5</td>
<td>4.61</td>
<td>0.034665</td>
<td>0.035098</td>
</tr>
<tr>
<td>5</td>
<td>4.65</td>
<td>0.034665</td>
<td>0.035098</td>
</tr>
<tr>
<td>10</td>
<td>1.5</td>
<td>0.000402</td>
<td>0.008239</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.001586</td>
<td>0.005475</td>
</tr>
<tr>
<td>10</td>
<td>2.5</td>
<td>0.001342</td>
<td>0.004882</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0.005639</td>
<td>0.006439</td>
</tr>
<tr>
<td>10</td>
<td>3.5</td>
<td>0.001856</td>
<td>0.004451</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.002803</td>
<td>0.004791</td>
</tr>
<tr>
<td>10</td>
<td>4.5</td>
<td>0.003958</td>
<td>0.004983</td>
</tr>
<tr>
<td>10</td>
<td>4.6</td>
<td>0.004075</td>
<td>0.005017</td>
</tr>
<tr>
<td>10</td>
<td>4.61</td>
<td>0.034665</td>
<td>0.035098</td>
</tr>
<tr>
<td>10</td>
<td>4.65</td>
<td>0.034665</td>
<td>0.035098</td>
</tr>
</tbody>
</table>

Results show that outliers removing allow to achieve better results than original ErrCor algorithm. Best testing MSE for Improved ErrCor have been achieved for OT=3.5 for both N=5 and N=10. Similarly, best training MSE for both N values have been achieved for the same value OT=1.5. This is because for lower value of OT much more patterns are removed during training process that causes better training. Table 2 shows the number of removed patterns during experiments. As can be observed for OT=5 number of removed patterns are higher than number of noised samples. Moreover, for best results with OT=3.5 number of removed patterns are lower than number of noised samples. Note, that for both values of N reaching the same value of OT=4.61 no outliers have been detected and removed that means results the same like for original ErrCor. Figure 3 shows training and testing process for original ErrCor. Training error is assigned by blue stars and testing error is assigned by red circles. It can be observed that best result is reached very quickly on the level of about 0.035 for both training and
Table 2. Number of removed patterns outliers

<table>
<thead>
<tr>
<th>OT</th>
<th>Neurons in network</th>
<th>Outliers for N=5</th>
<th>Outliers for N=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>450</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>361</td>
<td>450</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>311</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>202</td>
<td>319</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>160</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>1484</td>
<td>769</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>224</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>197</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>188</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>151</td>
<td>199</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>130</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>890</td>
<td>423</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>71</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>121</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>89</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>123</td>
<td>145</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>111</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>515</td>
<td>248</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>29</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>41</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>60</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>105</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>93</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>309</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>19</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>28</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>82</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>79</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>211</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>85</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>92</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>204</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>38</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>55</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>90</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>197</td>
<td>23</td>
<td></td>
</tr>
</tbody>
</table>
testing datasets. Figures 4-7 show training process for selected values of OT and both analyzed N. As can be observed, the training error is changing abruptly with removing of patterns while testing error is decreasing rather slowly but in the wake of changes of training error. The interesting results have been achieved for OT = 3 where both training and validating errors are relatively high and very close to each other. It means that for some values of OT training process can falls into local minimum and is not able to reach better results. This is especially visible in the case of N=5, where achieving result is not significantly better than for original ErrCor. In this case only 29 outliers have been removed during training process that was too small to eliminate noised patterns. On the second case, for N=10, better results have been obtained only for larger RBF network reaching testing MSE = 0.004294 and training MSE as low as 0.000046.

Fig. 3. The process of training with original ErrCor

Fig. 4. The learning process modified algorithm ErrCor: a) OT=1.5, N=5, b) OT=1.5, N=10
In the second experiment have been used real world datasets from UCI Machine Learning Repository commonly used as a benchmarks, such as Airplane Delay, Machine CPU, Auto Price, California Housing. For each dataset results of original ErrCor algorithm has been compared to discussed modified ErrCor with parameters $OT=5$ and $N=5$. Results of these experiments are shown in Figure 8 and Figure 9. Again, blue stars for given number of RBF units are a training MSE, red circles are testing MSE. As can be observed outliers eliminating allows to reach better results for smaller number of units also for real world datasets.

5 Conclusions

The paper presents proposition of improvement for Error Correction algorithm by elimination of inconsistent patterns from training process. Achieved experimental results
Fig. 7. The learning process modified algorithm ErrCor: a) OT=3.5, N=5, b) OT=3.5, N=10

Fig. 8. Results achieved for Airplane Delay and Machine CPU datasets
confirm effectiveness of proposed method that was originally suggested in [21]. Mentioned effectiveness depends however on the content of processed dataset and will be higher for more noisy data with more random corrupted data, that will be easy eliminated. Further work in this area will be focused on improvement proposed approach by searching a way that allow to find optimal training parameters for given dataset, as well as applying presented method for other training algorithms such as ELM or NBN.

References

Core for Large Datasets: Rough Sets on FPGA

Maciej Kopczynski, Tomasz Grzes, and Jaroslaw Stepaniuk

Faculty of Computer Science
Bialystok University of Technology
Wiejska 45A, 15-351 Bialystok, Poland
{m.kopczynski,t.grzes,j.stepaniuk}@pb.edu.pl
http://www.wi.pb.edu.pl

Abstract. In this paper we propose the FPGA and softcore CPU based device for large datasets core calculation using rough set methods. Presented architecture has been tested on two real datasets by downloading and running presented solution inside FPGA. Tested datasets had 1 000 to 10 000 000 objects. The same operations were performed in software implementation. Obtained results show the big acceleration in computation time using hardware supporting core generation in comparison to pure software implementation.

Key words: Rough sets, FPGA, hardware, core

1 Introduction

The rough sets theory developed in the eighties of the twentieth century by Prof. Z. Pawlak is an useful tool for data analysis. Therefore a lot of rough sets algorithms were implemented in scientific and commercial tools for data processing. But data processing efficiency problem is arising with the increase of the amount of data. Software limitations led to searching new possibilities.

Field Programmable Gate Arrays (FPGAs) are the digital integrated circuits which functions can be programmed by engineer at any time. It gives the possibility of evaluating any boolean functions. That’s why they can be used for supporting rough sets calculations.

At the moment there are some hardware implementations of specific rough set methods. The idea of sample processor generating decision rules from decision tables was described in [8]. In [5] the authors presented architecture of rough set processor based on cellular networks described in [7]. In [1] a concept of hardware device capable of minimizing the large logic functions created on the basis of discernibility matrix was developed. More detailed summary of the existing ideas and hardware implementations of rough set methods can be found in [2] and in [13]. Previous authors’ research results focused on this subject can be found in [3, 4, 12].

The paper is organized as follows. In Section 2 some information about the notion of core and datasets used during research are provided. The Section 3 focuses on description of hardware solution, while Section 4 is devoted to the experimental results.
2  Introductory Information

2.1  The Notion of Core in the Rough Set Theory

In decision table some of the condition attributes may be superfluous (redundant in other words). This means that their removal cannot worsen the classification. The set of all indispensable condition attributes is called the core. None of its elements can be removed without affecting the classification power of all condition attributes. In order to compute the core we can use discernibility matrix. The core is the set of all single element entries of the discernibility matrix.

A much more detailed description of the concept of the core can be found, for example, in the article [9] or in the book [11].

2.2  Algorithm CORE-DDM for Generating Core Using Discernibility Matrix

Below one can find pseudocode for simple algorithm CORE-DDM (CORE Direct Discernibility Matrix) for calculating core using discernibility matrix. More detailed description of this approach can be found in [9, 11].

INPUT: discernibility matrix $DM$
OUTPUT: core $C \subseteq A$

1: $C \leftarrow \emptyset$
2: for $x \in U$ do
3:   for $y \in U$ do
4:     if $|DM(x, y)| = 1$ and $DM(x, y) \not\subset C$ then
5:       $C \leftarrow C \cup DM(x, y)$
6:     end if
7:   end for
8: end for

The main concept of algorithm CORE-DDM is based on a property of singleton i.e. cell from discernibility matrix consisted of the only one attribute. This property tells that any singleton cannot be removed without affecting the classification power.

Input for the algorithm is the discernibility matrix $DM$. Output is core $C$ as a subset of condition attributes set denoted as $A$. Core is initialized as empty set in line 1. Two loops in lines 2 and 3 iterates over all objects (denoted as $U$) in discernibility matrix. Condition instruction in line 4 checks if matrix cell contains only one attribute. If so, then this attribute is added to the core $C$.

2.3  Algorithm CORE-IDM for Generating Core with No Discernibility Matrix

The main disadvantage of using discernibility matrix for big datasets is its size. Memory complexity of creating this type of matrix is $|U|^2|A|$, where $A$ is the condition attributes set. This makes a simple solution showed in previous subsection unusable for big data. For example the table of 1,000,000 objects and 8 condition attributes encoded as single bits requires 1TB of memory.

We have proposed our algorithm CORE-IDM (CORE Indirect Discernibility Matrix) which uses discernibility table indirectly to perform comparison between each
row of given decision table. This approach, basing on CORE-DDM algorithm in principles, has been designed by authors of this paper. Below is the pseudocode for this algorithm:

**INPUT:** decision table $DT = (U, A \cup \{d\})$

**OUTPUT:** core $C \subseteq A$

1: $C \leftarrow \emptyset$
2: for $x \in U$ do
3:   for $y \in U$ do
4:     if $d(x) \neq d(y)$ then
5:       $count \leftarrow 0$
6:       for $a \in A$ do
7:         if $a(x) \neq a(y)$ then
8:           $count \leftarrow count + 1$
9:           $candidate \leftarrow a$
10:      end if
11:    end for
12:   if $count = 1$ and $candidate \notin C$ then
13:     $C \leftarrow C \cup \{candidate\}$
14:   end if
15: end for
16: end for

Input to the algorithm CORE-IDM is decision table $DT$, and output is core $C$. $A$ denotes condition attributes set. In the first step core $C$ is initialized as empty set. Two loops in lines 2 and 3 take subsequent objects from decision table for comparison. Line 4 performs the comparison between decision attribute value of two objects $x$ and $y$. If these two objects belong to different decision classes, the rest of the algorithm is processed. $count$ variable, responsible for storing the number of differences on condition attributes values between objects $x$ and $y$ is set to 0 in line 5. Loop in line 6 iterates over set of condition attributes $A$. Values of a condition attribute is compared between objects $x$ and $y$ in line 7. In case of difference, the $count$ variable is incremented and a attribute is stored in $candidate$ variable. When the attribute loop finishes, attribute in $candidate$ variable is added to the core if $count$ variable is equal to 1 and this attribute is not in core (lines 12 to 14).

2.4 Algorithm CORE-HIDM for Hardware Supported Core Calculation with No Discernibility Matrix

Algorithm described in previous section cannot be run in hardware because of FPGA resources limitations. It is impossible to store large dataset inside hardware module. This is the reason why we present the modification of previous algorithm - CORE-HIDM (CORE Hardware Indirect Discernibility Matrix). Main idea is to divide the entire dataset into parts stored in two independent memory units for hardware module. These parts are subsequently processed by the unit. Pseudocode for the algorithm is given below:
INPUT: decision table \( DT = (U, A \cup \{d\}) \)

OUTPUT: core \( C \subseteq A \)

1: \( C \leftarrow \emptyset \)
2: for \( \text{cnt}_1 \leftarrow 0 \) to \( m - 1 \) do
3: \( \text{RAM}_1 \leftarrow \{x \in U : x_{\text{cnt}_1 \cdot n} \text{ to } x_{(\text{cnt}_1 + 1) \cdot n - 1}\} \)
4: for \( \text{cnt}_2 \leftarrow \text{cnt}_1 \) to \( m - 1 \) do
5: \( \text{RAM}_2 \leftarrow \{x \in U : x_{\text{cnt}_2 \cdot n} \text{ to } x_{(\text{cnt}_2 + 1) \cdot n - 1}\} \)
6: for \( x \in \text{RAM}_1 \) do
7: for \( y \in \text{RAM}_2 \) do
8: if \( d(x) \neq d(y) \) then
9: \( \text{count} \leftarrow 0 \)
10: for \( a \in A \) do
11: if \( a(x) \neq a(y) \) then
12: \( \text{count} \leftarrow \text{count} + 1 \)
13: \( \text{candidate} \leftarrow a \)
14: end if
15: end for
16: if \( \text{count} = 1 \) and \( \text{candidate} \notin C \) then
17: \( C \leftarrow C \cup \{\text{candidate}\} \)
18: end if
19: end if
20: end for
21: end for
22: end for

Input to the algorithm CORE-HIDM is decision table \( DT \), and output is core \( C \). 
\( A \) denotes condition attributes set. In the first step core \( C \) is initialized as empty set. 
Two loops in lines 2 and 4 are responsible for choosing parts of input decision table. 
Decision table is divided into \( m \) parts, where each of them have the size of \( n \) objects. In lines 3 and 5 chosen parts are loaded into RAM memories of hardware unit. The rest of the algorithm (lines 6 ro 21) is similar to the previously described. The only difference is that the objects for comparison are loaded from RAM memories.

### 2.5 Data to Conduct Experimental Research

In this paper, we present the results of the conducted experiments using two datasets: 
Poker Hand Dataset (created by Robert Catral and Franz Oppacher) and data about 
children with insulin-dependent diabetes mellitus (type 1).

First dataset was obtained from UCI Machine Learning Repository [6]. Each of 1 
000 000 records is an example of a hand consisting of five playing cards drawn from a 
standard deck of 52. Each card is described using two attributes (suit and rank), for a 
total of 10 predictive attributes. There is one decision attribute that describes the "Poker 
Hand". Decision attribute describes 10 possible combinations of cards in descending 
probability in the dataset: nothing in hand, one pair, two pairs, three of a kind, straight, 
flush, full house, four of a kind, straight flush, royal flush.
Insulin-dependent diabetes mellitus is a chronic disease of the body’s metabolism characterized by an inability to produce enough insulin to process carbohydrates, fat, and protein efficiently. Treatment requires injections of insulin. Twelve condition attributes, which include the results of physical and laboratory examinations and one decision attribute (microalbuminuria) describe the database used in our experiments. The data collection so far consists of 107 cases. The database is shown at the end of the paper [10]. A detailed analysis of the above data (only with the use of software systems) is in chapter 6 of the book [11].

The Poker Hand database was used for creating smaller datasets consisting of 1 000 to 500 000 of objects by selecting given number of first rows of original dataset. Diabetes database was used for generating bigger datasets consisting of 1 000 to 10 000 000 of objects. New datasets were created by multiplying the rows of original dataset. Created datasets had to be transformed to binary version. Numerical values were discretized and each attributes’ value was encoded using four bits for both datasets. Every single object was described on 44 bits for Poker Hand and 52 bits for Diabetes. To fit to memory boundaries in both cases, objects descriptions had to be extended to 64 bits words filling unused attributes with binary 0’s.

3 Hardware Implementation

Solution created by the authors uses combination of softcore processor and hardware unit designed to calculate the core. Diagram of the device is shown on Fig. 1. Core calculation process for large data sets is based on algorithm CORE-HIDM described in Section 2.4. The same input size of the module was used for both datasets.

![Diagram of core calculation module for large datasets](image)

Fig. 1. Diagram of core calculation module for large datasets

Purpose of processor is to:

1. Control the process of dividing large input decision table.
2. Control the core hardware calculation block.
3. Reload the data between internal and external RAM memories.
4. Process the results returned by core hardware calculation block.

Selected processor is NIOS II. This is the proprietary softcore unit provided by Altera for its FPGAs. It is fully functional, 32-bit, RISC processor with support of modern solutions enhancing the calculation power (e.g. multi-stage pipeline, dynamic branch prediction, separate instruction and data cache, MMU, MPU, . . . ).

DDR2 memory stores the large input decision table. SD card is the temporary solution for transferring data from PC to FPGA based solution. Data from SD card is copied to FPGAs DDR2 memory in the beginning of calculation process.

Core calculation unit is responsible for hardware support related to calculating subcores for given parts of decision table. Authors have used modified version of sequential core hardware calculation unit described in paper [4]. This unit has been extended in order to allow processing large datasets. To give reader better overview of prepared solution, the previous sequential core module is shortly described below.

The architecture of the sequential hardware core calculation unit shown on Fig. 2. Input of this block is decision table. Circuit consists of five functionally separated blocks:

1. **Comparators** – block of identical comparators which calculate the single row of discernibility matrix.
2. **OR-gates cascade** – block of OR-gates connected in a cascade. Every gate calculates logical OR operation on two elements: one from previous gate in a cascade and second from comparator.
3. **Singleton Detector** – block for checking if single row in discernibility matrix is a singleton (contains only one logical ‘1’). Outputs from this block are connected to OR-gates cascade.
4. **Multiplexer MUX** – in every turn selects the following object from decision table and puts it into the comparators in order to calculate single row of discernibility matrix.
5. **Control Logic** – responsible for storing calculation data and controls overall operations of the module.

Discernibility matrix entries are calculated by comparators very quickly, mostly because of simplicity of each comparator architecture. Then all entries goes to OR-gates cascade. The time to calculate the result depends on the size of discernibility matrix, increasing with its size. Last gate in cascade stores the result of calculations in the **CORE** register.

Sequential type of the core calculation module limits the utilization of LEs (Logical Elements) in FPGA. The disadvantage of this unit is its processing speed. Number of cycles needed to complete the calculation is equal to the number of objects in the input decision table.

Core calculation unit described previously cannot handle bigger data than size of the input register related to decision table. In order to process large datasets two blocks of fast static RAM memories were added to the solution. The control unit of the module has also been modified. Block diagram of modified unit is presented on Fig. 3.
Fig. 2. Block diagram of the hardware implementation of sequential core calculation module

Fig. 3. Block diagram of the hardware implementation of sequential core calculation module with modifications for large datasets
RAM were created as instances of dedicated FPGA blocks (MLAB, M9k and M144k). MLAB blocks are synchronous, dual-port memories with configurable organisation 32 x 20 or 64 x 10. M9k and M144k blocks are synchronous, true dual-port memory blocks with registered inputs and optionally registered outputs with many possible configurable organisations. Both data memories used in module, denoted as RAM1 and RAM2, store parts of input decision table to comparison. In the beginning, memories contain the same part of decision table. When objects from RAM2 were compared with all objects from RAM1, then RAM2 is reloaded with next part of decision table, until decision table has any not compared elements in it. Then RAM1 and RAM2 are loaded with second part of dataset and whole process continues.

4 Experimental Results

For the research purpose the core calculation was implemented in C language. Algorithms CORE-DDM described in Section 2.2 and CORE-IDM described in Section 2.3 were used.

The results of the software implementation were obtained using a PC equipped with an 8 GB RAM and 4-core Intel Core i7 3632QM with maximum 3.2 GHz in Turbo mode clock speed running Windows 7 Professional operational system. The source code of application was compiled using the GNU GCC 4.8.1 compiler. Given times for smaller datasets are averaged for 1 000 runs of algorithm with the same input data.

Quartus II 13.1 was used for design, compilation, synthesis and verifying simulation of the hardware implementation in VHDL language. Synthesized hardware blocks were downloaded and run on TeraSIC DE-3 equipped with Stratix III EP3SL150F1152C2N FPGA chip. FPGA clock running at 50 MHz for the sequential parts of the project was derived from development board oscillator. Implemented algorithm CORE-HIDM is presented in Section 2.4.

NIOS II softcore processor, as well as most parts of embedded system were instantiated using Qsys 13.1 tool. Software for NIOS II was implemented in C language using NIOS II Software Build Tools for Eclipse IDE.

Timing results were obtained using LeCroy waveSurfer 104MXs-B (1 GHz bandwidth, 10 GS/s) oscilloscope. For longer times, hardware time measurement units instantiated inside FPGA were used.

It should be noticed, that PCs clock is \( \frac{clk_{PC}}{clk_{FPGA}} = 64 \) times faster than development boards clock source.

All calculations were performed using datasets described in Section 2.5 with sizes between 1 000 and 10 000 000 objects.

Table 1 presents the results of the time elapsed for hardware and software solution using indirect row-by-row discernibility matrix calculation (algorithms CORE-IDM and CORE-HIDM described in Sections 2.3 and 2.4). Table 2 presents the results of the time elapsed for hardware and software solution using direct discernibility matrix calculation (software algorithm CORE-DDM and hardware algorithm CORE-HIDM described in Sections 2.2 and 2.4). Last two columns in both tables describe the speed-up factor without \( (C) \) and with \( (C_{clk}) \) taking clock speed difference between PC and FPGA into consideration. Abbreviations in objects number are: \( k = 10^3 \), \( M = 10^6 \).
Core generation related to software implementation using direct discernibility matrix calculation could not be performed for datasets having more than 10,000 objects because of extensive memory usage which is \( n^{2k} \), where \( k \) denotes number of conditional attributes and \( n \) is the number of objects in decision table. One must remember that each cell of matrix is described by 32 bit value using linked list consisting of another 32 bit values, what results in extensive memory usage.

Table 1. Comparison of execution time between hardware (algorithm CORE-HIDM) and software (algorithm CORE-IDM) implementation without using discernibility matrix explicit calculation for both datasets

<table>
<thead>
<tr>
<th>Objects</th>
<th>Hardware (- t_H )</th>
<th>Software (- t_S )</th>
<th>( C = \frac{t_S}{t_H} )</th>
<th>( C_{clk} = 64\frac{t_S}{t_H} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[s]</td>
<td>[s]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Poker Hand dataset</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1k</td>
<td>0.003</td>
<td>0.033</td>
<td>10.875</td>
<td>695.992</td>
</tr>
<tr>
<td>2.5k</td>
<td>0.013</td>
<td>0.143</td>
<td>11.119</td>
<td>711.632</td>
</tr>
<tr>
<td>5k</td>
<td>0.055</td>
<td>0.603</td>
<td>10.951</td>
<td>700.876</td>
</tr>
<tr>
<td>10k</td>
<td>0.207</td>
<td>2.410</td>
<td>11.623</td>
<td>743.851</td>
</tr>
<tr>
<td>25k</td>
<td>1.225</td>
<td>14.721</td>
<td>12.015</td>
<td>768.940</td>
</tr>
<tr>
<td>50k</td>
<td>4.710</td>
<td>58.726</td>
<td>12.469</td>
<td>798.043</td>
</tr>
<tr>
<td>100k</td>
<td>21.737</td>
<td>237.942</td>
<td>10.946</td>
<td>700.572</td>
</tr>
<tr>
<td>250k</td>
<td>130.947</td>
<td>1,515.449</td>
<td>11.573</td>
<td>740.674</td>
</tr>
<tr>
<td>500k</td>
<td>506.225</td>
<td>6,092.916</td>
<td>12.036</td>
<td>770.302</td>
</tr>
<tr>
<td>1M</td>
<td>1,850.523</td>
<td>24,313.094</td>
<td>13.138</td>
<td>840.864</td>
</tr>
<tr>
<td><strong>Diabetes dataset</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1k</td>
<td>0.003</td>
<td>0.018</td>
<td>5.911</td>
<td>378.325</td>
</tr>
<tr>
<td>2.5k</td>
<td>0.013</td>
<td>0.078</td>
<td>6.044</td>
<td>386.827</td>
</tr>
<tr>
<td>5k</td>
<td>0.055</td>
<td>0.328</td>
<td>5.953</td>
<td>380.980</td>
</tr>
<tr>
<td>10k</td>
<td>0.207</td>
<td>1.31</td>
<td>6.318</td>
<td>404.340</td>
</tr>
<tr>
<td>25k</td>
<td>1.225</td>
<td>8.002</td>
<td>6.531</td>
<td>417.978</td>
</tr>
<tr>
<td>50k</td>
<td>4.710</td>
<td>34.216</td>
<td>7.265</td>
<td>464.970</td>
</tr>
<tr>
<td>100k</td>
<td>21.737</td>
<td>135.309</td>
<td>6.225</td>
<td>398.390</td>
</tr>
<tr>
<td>250k</td>
<td>130.947</td>
<td>861.781</td>
<td>6.581</td>
<td>421.194</td>
</tr>
<tr>
<td>500k</td>
<td>506.225</td>
<td>3,464.821</td>
<td>6.844</td>
<td>438.043</td>
</tr>
<tr>
<td>1M</td>
<td>1,850.523</td>
<td>13,825.976</td>
<td>7.471</td>
<td>478.169</td>
</tr>
<tr>
<td>2.5M</td>
<td>11,729.536</td>
<td>81,234.655</td>
<td>6.926</td>
<td>443.242</td>
</tr>
<tr>
<td>5M</td>
<td>46,392.722</td>
<td>331,931.342</td>
<td>7.155</td>
<td>457.908</td>
</tr>
<tr>
<td>10M</td>
<td>185,182.545</td>
<td>1,293,245.545</td>
<td>6.984</td>
<td>446.952</td>
</tr>
</tbody>
</table>

FPGA resources utilization is fixed and is independent of the input dataset size. Datasets are divided into parts which are processed by the module. Module uses 21,562 Logical Elements (LE) of 113,600 total available.
Table 2. Comparison of execution time between hardware (algorithm CORE-HIDM) and software (algorithm CORE-DDM) implementation using explicit discernibility matrix calculation for both datasets

<table>
<thead>
<tr>
<th>Objects</th>
<th>Hardware - $t_H$ [s]</th>
<th>Software - $t_S$ [s]</th>
<th>$C = \frac{t_S}{t_H}$</th>
<th>$C_{clk} = 64 \frac{t_S}{t_H}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Poker Hand dataset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1k</td>
<td>0.003</td>
<td>0.296</td>
<td>97.209</td>
<td>6 221.346</td>
</tr>
<tr>
<td>2.5k</td>
<td>0.013</td>
<td>1.843</td>
<td>142.813</td>
<td>9 140.023</td>
</tr>
<tr>
<td>5k</td>
<td>0.055</td>
<td>7.496</td>
<td>136.044</td>
<td>8 706.788</td>
</tr>
<tr>
<td>&gt;10k</td>
<td>0.207</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Diabetes dataset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1k</td>
<td>0.003</td>
<td>0.154</td>
<td>50.575</td>
<td>3 236.782</td>
</tr>
<tr>
<td>2.5k</td>
<td>0.013</td>
<td>1.096</td>
<td>84.928</td>
<td>5 435.413</td>
</tr>
<tr>
<td>5k</td>
<td>0.055</td>
<td>5.864</td>
<td>106.425</td>
<td>6 811.180</td>
</tr>
<tr>
<td>10k</td>
<td>0.207</td>
<td>29.021</td>
<td>139.961</td>
<td>8 957.531</td>
</tr>
<tr>
<td>&gt;25k</td>
<td>1.225</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Fig. 4 presents a graphs showing the relationship between the number of objects and execution time for hardware and software solution using indirect discernibility matrix calculation for both datasets. Both axes have the logarithmic scale.

Presented results show big increase in the speed of data processing. Hardware module execution time compared to the software implementation using row-by-row discernibility matrix calculation is 5 to 12 times faster, while comparing time to need to obtain full discernibility matrix is around 50 to 142 times faster in case of biggest processed dataset. If we take clock speed difference between PC and FPGA under consideration, these results are much better - average speed-up factor is 378 to 840 for indirect discernibility matrix method and up to 9 140 for method using discernibility matrix calculation first. The hardware core calculation unit was not optimized. Results are expected to be few times better after optimization.

We have used the same module size (configuration) for both datasets, that is why time results for the hardware are the same. It is not important what type of data is processed for hardware unit.

Let comparison of attribute value between two objects or retrieving the element from discernibility matrix be an elementary operation. $k$ denotes number of conditional attributes and $n$ is the number of objects in decision table. Computational complexity of software implementation for the core calculation is $\Theta(kn^2 + n^2)$ according to algorithm CORE-DDM in Section 2.2 (discernibility matrix calculation and core calculation). For algorithm CORE-IDM described in Section 2.3 it is $\Theta(kn^2)$. Using hardware implementation (CORE-HIDM), complexity of core calculation is $\Theta(n^2)$. The $k$ is missing, because our solution performs comparison between all attributes in $\Theta(1)$ - all attributes values between two objects are compared in single clock cycle. Additionally, core module performs comparisons between many objects at time. In most cases $k << n$, so we
Fig. 4. Relation between number of objects and calculation time for hardware (algorithm CORE-HIDM) and software implementation (algorithm CORE-IDM) using indirect discernibility matrix calculation method for both datasets.

can say, that computational complexity for software and hardware implementations are the same.

5 Conclusions and Future Research

Performing core calculations using hardware implementations gives us a big acceleration in comparison to software solution. If we compare the results to core calculation using discernibility matrix we can notice two advantages: shorter time to finish calculations and possibility of processing much bigger datasets.

Core hardware calculation unit was not optimized for performance in this paper. Processing time can be substantially reduced by increasing FPGA clock frequency and by modifying control unit to introduce triggering on both edges of clock signal. This will speed-up processing time nearly twice.

Hardware solution presented in this paper is easibly scalable. Duplicating calculation unit will improve processing speed. One must remember that this approach needs preparation of specialized control unit responsible for controlling performing concurrent operations (data flow of both input and output).

Another calculation speed impact factor is the size of hardware processing module in terms of capacity to process given number of objects. Bigger size of the unit will allow to shorten calculation time.

Further research will focus on checking different sizes of core module and obtaining results of performing the calculations in parallel by multiplying units.
Acknowledgements

The research is supported by the Polish National Science Centre under the grant 2012/07/B/ST6/01504 (Jarosław Stepaniuk) and by the scientific grant S/WI/3/2013 (Tomasz Grzes and Maciej Kopczynski).

References

Sequential P Systems with Active Membranes Working on Sets*

Michal Kováč and Damas P. Gruska
Faculty of Mathematics, Physics and Informatics, Comenius University

Abstract. We study variants of P systems that are working in the sequential mode. Usually, they are not computationally complete, but there are possible extensions that can increase the computation power. Extensions that implement a notion of zero-checking are often computationally complete. P systems with an ability to create new membranes are a rare exception as they are known to be computationally complete even in the sequential mode without using a dedicated zero-check operation. Using sets instead of multisets was inspired by Reaction systems and we show how to use this relaxation in the context of active membranes. We challenge the original definition of a membrane creation because possible multiplicity of labels of child membranes are in conflict with no multiplicity of objects in Reaction systems. We propose more suitable notions of membrane creation and prove computational completeness by simulating a register machine.

1 Introduction

Membrane systems (P systems) [1] were introduced by Păun (see [2]) as distributed parallel computing devices inspired by the structure and functionality of cells. Starting from the observation that there is an obvious parallelism in the cell biochemistry and relying on the assumption that “if we wait enough, then all reactions which may take place will take place”, a feature of the P systems is given by the maximal parallel way of using the rules. For various reasons, ranging from looking for more realistic models to just more mathematical challenge, the maximal parallelism was questioned, either simply criticized, or replaced with presumably less restrictive assumptions. In some cases, a sequential model may be a more reasonable assumption. In sequential P systems, only one rewriting rule is used in each step of computation. Without priorities, they are equivalent to Petri nets [3], hence not computationally complete. However, priorities, inhibitors and other modifications can increase the computation power. It seems that there is a link between universality and ability to zero-check [4].

Standard models of membrane systems have configurations, where any given compartment is represented as a multiset of objects and each computational action is represented by a multiset of simultaneously executed (multiple copies of) individual evolution rules.

Such strong reliance on counting (through multiple copies of objects and rules) may lead to potential problems in two respects. First, one may wonder how realistic is the counting (multiset) mechanism if one needs to represent huge numbers of molecules

* Work supported by the grant VEGA 1/1333/12.
and instances of biochemical reactions. Second, a membrane system would normally have an infinite state space, making the application of formal verification techniques impractical or indeed impossible (there exists a rich body of results proving Turing completeness of even very simple kinds of membrane systems).

A radical solution to the state space problems can be provided by reaction systems, which, however, model biochemical reactions in living cells using qualitative (based on presence and absence of entities) rather than quantitative rewriting rules.

Set membrane systems [5] are based on sets (of objects or rules) together with the associated set operations, rather than on multisets with multiset operations. An interesting property of maximal parallel steps in set membrane systems is that there is always exactly one maximal parallel set of simultaneously applicable rules, thus such system is deterministic.

Alhazov in [6] proposed P systems, where the multiplicities of objects are ignored, which is essentially the same as set membrane systems. He proved that with bounded number of membranes they have a very limited computing power, exactly the Parikh images of regular languages. On the other hand, allowing membrane creation or division implies the computational completeness.

The sequential mode was only mentioned in [5] under the notion of “min-enabled” computational step. As well as in the maximal parallel mode, the sequential set membrane systems can also generate only the regular languages [6].

In section 2 we recall some computer science basic notions that we will use through the work. Sequential P systems with active membranes working on sets instead of multisets (Sequential active set P systems) are formally presented in section 3.

In section 4 we show the computational completeness by simulating the register machine. In the last section we propose two modifications of the definition of a membrane creation because in the original definition possible multiplicity of labels of child membranes are in conflict with no multiplicity of objects. First modification is called inject-or-create and has no explicit membrane creation rule. Instead, when sending objects to a child membrane, which does not exist, then a new membrane is created and objects are sent to it. Second modification is called wrap-or-create and has an explicit membrane creation rule. However, applying such rule when a child membrane with the same label exists wraps the existing membrane in the newly created one.

2 Preliminaries

Here we recall several notions from the classical theory of formal languages.

An alphabet is a finite nonempty set of symbols. Usually, it is denoted by $\Sigma$. A string over an alphabet is a finite sequence of symbols from the alphabet. We denote by $\Sigma^*$ the set of all strings over an alphabet $\Sigma$. By $\Sigma^+ = \Sigma^* - \{\varepsilon\}$ we denote the set of all nonempty strings over $\Sigma$. A language over the alphabet $\Sigma$ is any subset of $\Sigma^*$.

The number of occurrences of a given symbol $a \in \Sigma$ in the string $w \in \Sigma^*$ is denoted by $|w|_a$. $\Psi_\Sigma(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$ is called a Parikh vector associated with the string $w \in \Sigma^*$, where $\Sigma = \{a_1, a_2, \ldots, a_n\}$. For a language $L \subseteq \Sigma^*$, $\Psi_\Sigma(L) = \{\Psi_\Sigma(w) | w \in L\}$ is the Parikh image of $L$. If FL is a family of languages, PsFL denotes the family of Parikh images of languages in FL.
Next, we recall notions from graph theory.

A **rooted tree** is a tree, in which a particular node is distinguished from the others and called the root node. Let $T$ be a rooted tree. We will denote its root node by $r_T$. Let $d$ be a node of $T \setminus \{r_T\}$. As $T$ is a tree, there is a unique path from $d$ to $r_T$. The node adjacent to $d$ on that path is also unique and is called a parent node of $d$ and is denoted by $\text{parent}_T(d)$. We will denote the set of nodes of $T$ by $V(T)$ and set of its edges by $E(T)$. Let $T_1, T_2$ be rooted trees. A bijection $f : V(T_1) \to V(T_2)$ is an isomorphism iff $\{(f(u), f(v)) | (u, v) \in E(T_1)\} = E(T_2)$ and $f(r_{T_1}) = r_{T_2}$.

### 3 Sequential Active Set P Systems

The fundamental ingredient of a P system is the membrane structure (see [7]). It is a hierarchically arranged set of membranes, all contained in the skin membrane. Each membrane determines a compartment, also called region, which is the space delimited from above by it and from below by the membranes placed directly inside, if any exists. Clearly, the correspondence between membranes and regions is one-to-one, that is why we sometimes use interchangeably these terms. The membrane structure can be also viewed as a rooted tree with the skin membrane as the root node.

A P system consists of a membrane structure, where each membrane is labeled with a number from 1 to $m$. Each membrane contains a set of objects. Objects can be transformed into other objects and sent through a membrane according to given rules defined for membrane labels. The rules are known from the beginning for each possible membrane, even for the ones that do not exist yet, or the ones that will never exist.

In this paper we work with sequential P systems with active membranes working on sets (Sequential active set P systems). The rules can modify the membrane structure by dissolving and creating new membranes. That is why we will define the configuration to include the membrane structure as well.

Let $\Sigma$ be a set of objects. A **membrane configuration** is a tuple $(T, l, c)$, where:

- $T$ is a rooted tree,
- $l \in \mathbb{N}^{V(T)}$ is a mapping that assigns for each node of $T$ a number (label), where $l(r_T) = 1$, so the skin membrane is always labeled with 1,
- $c \in (2^\Sigma)^{V(T)}$ is a mapping that assigns for each node of $T$ a set of objects from $\Sigma$, so it represents the contents of the membrane.

The common representation of a membrane structure in this paper is by a string, where a membrane is denoted by a pair of matching square brackets, e.g. $[1[2ab][2ac]]_1$.

A **sequential active set P system** is a tuple $(\Sigma, C_0, R_1, R_2, \ldots, R_m)$, where:

- $\Sigma$ is a set of objects,
- $C_0$ is the initial membrane configuration,
- $R_1, R_2, \ldots, R_m$ are finite sets of rewriting rules associated with the labels 1, 2, \ldots, $m$ and can be of forms:
  - $u \to w$, where $u \subseteq \Sigma, |u| \geq 1, w \subseteq (\Sigma \times \{\cdot, \uparrow, \downarrow_j\})$ and $1 \leq j \leq m$,
  - a dissolving rule $u \to w\delta$, where $u \subseteq \Sigma, |u| \geq 1, w \subseteq (\Sigma \times \{\cdot, \uparrow, \downarrow_j\})$ and $1 \leq j \leq m$. 


For the first two forms, each rewriting rule may specify for each object on the right side, whether it stays in the current region (we will omit the symbol \( \cdot \)), moves through the membrane to the parent region (\( \uparrow \)) or to a specific child region (\( \downarrow j \), where \( j \) is a label of a membrane). We denote these transfers with an arrow immediately after the symbol. An example of such rule is the following: \( ab \rightarrow ab \downarrow 2 c \uparrow c\delta \).

By applying the rule we mean the removal of objects specified on the left side and the addition of the objects on the right side with respect to set union semantics. Symbol \( \delta / \notin \Sigma \) does not represent an object. It may be present only at the end of the rule, which means that after the application of the rule, the membrane is dissolved and its contents (objects, child membranes) are propagated to the parent membrane.

Active P systems differ from classic (passive) P systems in their ability to create new membranes by rules of the third form. Such rule will create new child membrane with a given label \( j \) and a given set of objects \( v_1 \) as its contents, while the set \( v_2 \) is the set of products that stays in the current membrane. If the current membrane already contains a child membrane with label \( j \), then such rule is not applicable.

For a sequential active set P system \((\Sigma, C_0, R_1, R_2, \ldots, R_m)\), configuration \( C = (T, l, c) \), membrane \( d \in V(T) \) the rule \( r \in R_{l(d)} \) is applicable iff:

- \( r = u \rightarrow w \) and \( u \subseteq c(d) \) and for all \( (a, \downarrow k) \in w \) there exists \( d_2 \in V(T) \) such that \( l(d_2) = k \land parent(d_2) = d \),
- \( r = u \rightarrow w\delta \) and \( u \subseteq c(d) \) and for all \( (a, \downarrow k) \in w \) there exists \( d_2 \in V(T) \) such that \( l(d_2) = k \land parent(d_2) = d \) and \( d \neq r_T \),
- \( r = u \rightarrow [jv_1]jv_2 \) and \( u \subseteq c(d) \).

In this paper we assume only sequential systems, so in each step of the computation, there is one rule nondeterministically chosen among all applicable rules in all membranes to be applied.

A computation step of a sequential active P system is a relation \( \Rightarrow \) on the set of membrane configurations such that \( C_1 \Rightarrow C_2 \) holds iff there is an applicable rule in a membrane in \( C_1 \), such that applying that rule can result in \( C_2 \).

The P system can work in generating or in accepting mode. For the generating mode we consider the concatentation of the objects which leave the system, in the order they are sent out of the skin membrane (if several symbols are expelled at the same time, then any ordering of them is considered). In this case we generate a language. The result of a single computation is clearly only one multiset or a string, but for one initial configuration there can be multiple possible computations. It follows from the fact that there can be more than one applicable rule in each configuration and they are chosen nondeterministically.

For the accepting mode the input word is encoded into a membrane structure by a given encoding and it is accepted if and only if a given accepting configuration can be reached[3].
4 Simulation of Register Machine

4.1 Register Machine

In this section we will show that sequential active set P systems are powerful computing devices as they can simulate the register machine. The section starts with a definition of deterministic register machine with a definition of a configuration. Next is the formulation of the main theorem followed by a proof made by a simulation. At last the efficiency of the simulation is questioned and various improvements are proposed.

Definition 1. A \( n \)-register machine is a tuple \( M = (n, P, i, h) \), where:

- \( n \) is the number of registers,
- \( P \) is a set of labeled instructions of the form \( j : (op(r), k, l) \), where \( op(r) \) is an operation on register \( r \leq n \), and \( j, k, l \) are labels from the set \( Lab(M) \) such that there are no two instructions with the same label \( j \),
- \( i \) is the initial label, and
- \( h \) is the final label.

The machine is capable of the following instructions:

- \((add(r), k, l)\) : Add one to the contents of register \( r \) and proceed to instruction \( k \) or to instruction \( l \); in the deterministic variants usually considered in the literature we demand \( k = l \).
- \((sub(r), k, l)\) : If register \( r \) is not empty, then subtract one from its contents and go to instruction \( k \), otherwise proceed to instruction \( l \).
- \textit{halt} : This instruction stops the machine. This additional instruction can only be assigned to the final label \( h \).

We will denote by \((op(r), _, _)\) any of the operations \textit{add} and \textit{sub} operating on the register \( r \) having arbitrary following instruction.

A deterministic \( m \)-register machine can analyze an input \((n_1, \ldots, n_m) \in \mathbb{N}_0^m\) in registers 1 to \( m \), which is recognized if the register machine finally stops by the halt instruction with all its registers being empty (this last requirement is not necessary). If the machine does not halt, the analysis was not successful.

A configuration of a register machine is a tuple \((r_1, \ldots, r_m, ip)\), where \( r_i \) is the value of the register \( i \) and \( ip \) (instruction pointer) is the label of current instruction to be executed.

4.2 Simple Simulation

The main theorem is stated as follows:

Theorem 1. Sequential active set P systems are computationally complete.

Proof. Computational completeness is proved by a direct simulation of a register machine, which is also computationally complete.
For a register machine with \( m \) registers we will construct a sequential active set \( \mathbb{P} \) system \((\Sigma, C_0, R_1, \ldots R_{m+1})\), where
\[
\Sigma = \{x_j, y_j \text{ for instructions with label } j\} \cup \{t_i \text{ for each register } i\}
\]
Skin membrane will be labeled with \( m+1 \), other labels correspond to registers 1 to \( m \). \( C_0 \) will be the input word for the register machine encoded into a membrane structure by the following encoding:

For a configuration of register machine \((r_1, r_2, \ldots r_m, ip)\) the membrane structure will consist of a skin membrane, which will contain \( m \) chains consisting of \( r_i \) membranes embedded one into another like in a Matryoshka doll with label \( i \). The innermost membranes will contain a single object \( t_i \). If \( r_i = 0 \) then \( t_i \) is in the skin membrane and there is no membrane with label \( i \). Object representing the label of the current instruction \((x_{ip})\) is in the skin membrane.

We will have following rules in the skin membrane:

1. \( y_j \rightarrow x_j \),
2. \( x_j \rightarrow x_j \uparrow_i \) for instruction \( j : (op(i), _, _) \),
3. \( x_j, t_i \rightarrow [y_k, t_i]_1 \) for instruction \( j : (add(i), k, k) \),
4. \( x_j, t_i \rightarrow l \) for instruction \( j : (sub(i), _, l) \)

For the membrane \( i \):

5. \( x_j \rightarrow x_j \downarrow_i \) for instruction \( j : (op(i), _, _) \),
6. \( x_j, t_i \rightarrow [y_k, t_i]_1 \) for instruction \( j : (add(i), k, k) \),
7. \( y_j \rightarrow y_j \uparrow \) for instruction \( j : (op(i), _, _) \),
8. \( x_j, t_i \rightarrow y_k, t_i, \delta \) for instruction \( j : (sub(i), k, l) \)

Object \( x_j \) represents the instruction currently executed. It is sent down the chain of membranes by rules 2 and 5. In the innermost membrane the creation of a new membrane (rule 6), or the dissolution (rule 8) is performed. Then the next instruction represented by object \( y_j \) is sent upwards all the way to the skin membrane by the rule 7. The object \( t_i \) is always present in the innermost membrane. For a SUB instruction there are two rules in the skin membrane, together they implement the zero-test. The rule 2 is applicable only if the register is nonempty and the rule 4 is applicable only if the register is empty by requiring the presence of \( t_i \), meaning that the value of register \( i \) is zero.

**Example 1.** Assume a register machine with two registers with values \( r_1 = 2 \), \( r_2 = 0 \) and the current instruction \( j : add(1, k, k) \). The corresponding membrane configuration is \([5][1[t_1]_1][x_j t_2]_3\). The computation of the \( \mathbb{P} \) system is deterministic, at each step there is only one applicable rule. Starting with the rule 2 and then the rule 5, \( x_j \) enters the innermost membrane, where the rule 6 creates new membrane.

The simulation was quite straightforward. We proved that the model is computationally complete. However, the simulation is not very effective. It uses alphabet of size \( 2 \ast \text{number of instructions} + \text{number of registers} \), and its number of membranes is linearly dependent on the sum of values of registers. The time needed for executing an instruction on register \( i \) is linearly dependent on \( r_i \).
4.3 Optimization of the Simulation

In this subsection we address the inefficient usage of membranes in the previous simulation. New, optimized simulation will reduce it to logarithmic dependency.

For a register machine with \( m \) registers we will construct a sequential active set \( P \) system, where \( \Sigma = \{0, 1, p, s, t\} \cup \{x_j, y_j, z_j\} \) for instructions with label \( j \). Skin membrane will be labeled with \( m + 1 \), other labels correspond to registers 1 to \( m \).

Assume configuration of register machine \( (r_1, r_2, \ldots r_m, ip) \). For each register \( i \), let \( b_1b_2\ldots b_k \) be a binary representation of \( r_i \). The skin membrane will contain a chain of \( k \) membranes embedded one into another like in a Matryoshka doll with label \( i \). The membrane in depth \( d \) will contain the object \( b_{k-d} \), which is either 0 or 1. So the highest-order position in the binary number is represented by the innermost membrane and more often changed positions by increments are in membranes closer to the skin membrane. Moreover, the innermost membranes contain a single object \( t \). The skin membrane contains the label of the current instruction \( x_{ip} \). Other membranes (not skin and not innermost) contain \( s \). Object \( p \) will be in all membranes except the skin membrane and direct children of the skin membrane. It represents the fact that the membrane can be dissolved, while keeping at least one membrane for binary representation of the register value.

The basic idea is to recursively decide the next action based on lowest position. For incrementing number ending with zero, incrementing the lowest position is enough. Similar simple case is when decrementing number ending with one. For incrementing number ending with one, we decrement the lowest position and recursively call increment on the binary number omitting the lowest position. Similarly, for decrementing number ending with zero, we increment the lowest position and recursively call decrement on the binary number omitting the lowest position. There are some special cases, like incrementing \( 111 \) to \( 1000 \) or decrementing \( 1000 \) to \( 111 \). In these cases we should change the number of membranes representing positions.

We will have following rules in the skin membrane:

1. \( y_j \rightarrow x_j \),
2. \( x_j \rightarrow x_j \downarrow_i \) for instruction \( j : \text{op}(i,_,_0) \)

For the membrane \( i \) and instruction \( j \):

3. \( y_j \rightarrow y_j \uparrow \) (return the next instruction to the skin membrane).

For the membrane \( i \) and instruction \( j : \text{add}(i,k,k) \):

4. \( x_j1 \rightarrow x_j \downarrow_i 0 \) (we decremented lower position, so we must increment higher position (011 to 100, now at 1 to 0)),
5. \( x_j0 \rightarrow y_k \uparrow 1 \) (we incremented a position and can return and proceed to the next instruction),
6. \( x_j1t \rightarrow [t1p]i; y_k \uparrow 0s \) (incrementing 111 to 1000).

For the membrane \( i \) and instruction \( j : \text{sub}(i,k,l) \):

7. \( x_j1s \rightarrow y_k \uparrow 0s \) (we found position to decrement, proceed to the next instruction),
8. \( x_j0 \rightarrow x_j \downarrow_i 1 \) (1000 is decremented to 0111 and now we encountered a 0),
9. $x_j1tp \rightarrow z_kt\delta$ (incrementing the number of bits),
10. $z_jst \rightarrow y_jt$ (after incremented the number of bits, remove $s$ in the new highest-order position),
11. $x_j0t \rightarrow y_l \uparrow 0t$ (trying to decrement a zero)

**Example 2.** Assume a register machine with two registers with values $r_1 = 3$, $r_2 = 0$ and the current instruction $j: \text{add}(1, k, k)$. The corresponding membrane configuration is $[3[11tp]_11s]_1[20t]_2x_j)_3$. The computation of the P system is deterministic, at each step there is only one applicable rule. Starting with the rule 2, $x_j$ will meet the object 1 in the configuration $[3[11tp]_11x_j]s]_1[20t]_2)_3$. The only applicable rule then is 4, resulting in $[3[11tpx_j]_10s]_1[20t]_2)_3$. $x_j$ now meets objects 1 and $t$, which means that the only applicable rule is 6, creating a new innermost membrane and resulting in the configuration $[3[11tp]_10y_k]s]_1[20t]_2)_3$. The object $y_k$ is by the rule 3 propagated to the skin membrane, where it is prepared for the next instruction by the rule 1.

One instruction of the register machine is performed by number of computational steps which is logarithmic on the value of the register the instruction is operated on. The number of membranes is logarithmic as well. The number of objects is $3 \times \text{number of instructions} + 5$.

### 4.4 Further Optimalizations

Could the simulation be optimized even more? Encoding the register value to a chain of membranes is not making full use of membrane structure. There are many options for a representation of an integer by a tree. For efficient implementation of the increment and decrement instructions, we need an encoding with a property that a local change in the value of the encoding of the entire tree corresponds to a local change in the value of the encodings of its child subtrees. Stein in 1999 [8] proposed a boustrophedonic variant of Cantor pairing function. The implementation of a sequential active set P system simulating a register machine using this pairing function to encode child subtrees would be quite easy, but we would stick to the logarithmic time in the worst case (diagonal of the pairing function). Catalan pairing function [9] orders full binary trees by the number of nodes. The time would be logarithmic with a base 4, which is a slight improvement, but asymptotically still the same.

### 5 Modified Membrane Creation Semantics

In this section we will investigate the effect of other semantics of membrane creation. The previous semantics assumed an explicit membrane creation rule. If the current membrane already contains child membrane with the same label as the membrane about to be created, then the rule is not applicable, and the membrane creation is aborted. Similar behavior is in the definition of sending objects to the child membrane. If such membrane does not exist, objects cannot be sent and the rule is not applicable.

These two behaviors are in fact complementary. It seems natural to join these two artificial rule abortions and provide a rule that will always be applicable if the precondition of left side inclusion is fulfilled.
5.1 Semantics Inject-or-Create

We will first examine the case where we have no explicit membrane creation rule. Any rule which is sending some objects to child membrane labeled \( j \) will create child membrane \( j \) if it does not exist.

Formally, rules can be of form:

- \( u \rightarrow w \), where \( u \subseteq \Sigma, |u| \geq 1 \), \( w \subseteq (\Sigma \times \{\cdot, ↑, ↓_{j}\}) \) and \( 1 \leq j \leq m \).
- a dissolving rule \( u \rightarrow w\delta \), where \( u \subseteq \Sigma, |u| \geq 1 \), \( w \subseteq (\Sigma \times \{\cdot, ↑, ↓_{j}\}) \) and \( 1 \leq j \leq m \).

For a sequential active set P system \((\Sigma, C_0, R_1, R_2, \ldots, R_m)\), configuration \( C = (T, l, c) \), membrane \( d \in V(T) \) the rule \( r \in R_{l(d)} \) is applicable iff:

- \( r = u \rightarrow w \) and \( u \subseteq c(d) \),
- \( r = u \rightarrow w\delta \) and \( u \subseteq c(d) \) and \( d \neq r_T \).

Example 3. Assume the membrane configuration \([1[2]a]\). If we apply the rule \( a \rightarrow a↓_{2} \) in the skin membrane, the object \( a \) is sent to the membrane 2 because such child membrane already exists. The resulting membrane configuration is \([1[2]a]\).

If we apply the rule \( a \rightarrow a↓_{3} \) in the skin membrane, a new membrane labeled 3 is created, because such child membrane does not exist yet. The resulting membrane configuration is \([1[2][3]a]\).

Theorem 2. Sequential active set P systems with inject-or-create semantics are computationally complete.

Proof. The simulation is essentially the same as in section 4.3. All the rules which are sending objects into a child membrane are already assuming that the child membrane already exists. The only difference is in the rule for membrane creation: \( x_jl_t \rightarrow [i1t_{p}]_{y{k}} ↑ 0s \). This rule is applied always in the innermost membrane with no child membranes. Modified simulation will therefore use rule \( x_jl_t \rightarrow 1↓_{i} t ↓_{i} p ↓_{i} y{k} ↑ 0s \), which, when applied, creates a child membrane \( i \), because no such child membrane exists. \( \square \)

The efficiency of this simulation is essentially the same as in section 4.3, that means logarithmic number of steps for simulating one instruction of the register machine as well as logarithmic number of membranes and the number of objects is \( 3 \ast \text{number of instructions} + 5 \).

5.2 Semantics Wrap-or-Create

In this variant we stay with explicit membrane creation rule, but when the membrane with the same label is already contained in the current membrane, the rule remains applicable and the child membrane will be wrapped by a new membrane with the given contents. For example, applying the rule \( a \rightarrow [2b]_{2}c \) in the membrane 1 of membrane structure \([1a[2d]_{2}] \) would result in \([1c[2b[2d]_{2}]_{2}] \).
Theorem 3. Sequential active set P systems with wrap-or-create semantics are computationally complete.

Proof. Again, we will show how to simulate the register machine. The simulation will be similar to the one defined in subsection 4.2, but with additional control objects similar to the second simulation 4.3.

For a register machine with \( m \) registers we will construct a sequential active set P system \( (\Sigma, C_0, R_1, \ldots, R_{m+1}) \), where

\[
\Sigma = \{ x_j \text{ for instructions with label } j \} \cup \{ t_i, s_i \text{ for each register } i \}
\]

Skin membrane will be labeled with \( m + 1 \), other labels correspond to registers 1 to \( m \). \( C_0 \) will be the input word for the register machine encoded into a membrane structure by the following encoding:

For a configuration of register machine \( (r_1, r_2, \ldots, r_m, t_p) \) the membrane structure will consist of a skin membrane, which will contain \( m \) chains consisting of \( r_i \) membranes embedded one into another like in a Matryoshka doll with label \( i \). Membranes with a child labeled \( i \) will contain a single object \( s_i \). If the membrane has no child labeled \( i \), it contains an object \( t_i \). If \( r_i = 0 \) then \( t_i \) is in the skin membrane and there is no membrane with label \( i \). Object representing the label of the current instruction \((x_{ip})\) is in the skin membrane.

We will have following rules in the skin membrane:

1. \( x_j s_i \rightarrow [s_i]_1 s_i x_k \) for instruction \( j : \text{(add}(i), k, \_\text{)} \),
2. \( x_j t_i \rightarrow [t_i]_1 s_i x_k \) for instruction \( j : \text{(add}(i), k, \_\text{)} \),
3. \( x_j t_i \rightarrow x t_i \) for instruction \( j : \text{(sub}(i), k, l) \),
4. \( x_j s_i \rightarrow x \downarrow_{l_i} \) for instruction \( j : \text{(sub}(i), k, l) \). For the membrane \( i \):
5. \( x_j \rightarrow x_k \delta \)

For every add instruction there is just one rule applied in the simulation and for each sub instruction there is one or two instructions, depending on the register value. If \( r_i > 0 \) then the instruction enters the membrane labeled \( i \) and dissolves it, decreasing the number of stacked membranes with label \( i \).

Example 4. Assume a register machine with two registers with values \( r_1 = 2 \) and \( r_2 = 0 \). The corresponding membrane configuration is \([s_1[t_1]1s_1]_1s_1t_2x_j]_3\]. If the current instruction is \( j : \text{sub}(1, k, l) \) then the only applicable is the rule 4 which results in the configuration \([s_1[t_1]1s_1x_j]_1s_1t_2]_3\]. Then the only applicable is the rule 5 which dissolves the membrane \( 1 \) resulting in the configuration \([s_1[t_1]1s_1]_1s_1t_2x_k]_3\].

For another example, consider instruction \( j : \text{add}(1, k, k) \). The increment is simulated by wrapping the membrane \( 1 \) to a new membrane created by the rule 1 with the resulting configuration \([s_1[t_1]1s_1]_1s_1]_1s_1t_2x_k]_3\].

This simulation is the most suitable for simulating the register machine because for incrementing the number of stacked membranes we just need to wrap the topmost membrane into a new membrane. This gained us constant time for executing one instruction, however the number of membranes remains linear on the sum of register values. The number of objects is number of instructions + 2 * number of registers.
6 Conclusions

We have investigated the bridge between membrane systems and reaction systems for the sequential variant with active membranes. We have shown computational completeness by a simulation of a register machine. When using sets instead of multisets, the original definition of creating a membrane may seem obsolete. Therefore, we have proposed alternative definitions for membrane creation: inject-or-create and wrap-or-create. In either case the resulting system has been shown to be universal.

As some simulations are not very effective, we have also proposed ways to improve the efficiency. For the simulation with original membrane creation we managed to reduce time needed for executing one instruction of the register machine from linear to logarithmic time. The wrap-or-create semantics is the most suitable for the simulation as for every instruction of the register machine only constant number of steps of the P system is needed.

The register value in the simulations is encoded in either unary or binary form. We propose other options to encode the register value as a tree structure of membranes and leave the construction of the simulation as a topic for further study.

References

Author Index

Akhundov, Jafar, 22
Aisolami, Fawaz, 4, 12
Amin, Talha, 4, 12
Barbuti, Roberto, 31
Barylska, Kamila, 44
Bazan, Jan G., 115
Bazan-Socha, Stanisława, 115
Benke, Marcin, 57
Buregwa-Czuma, Sylwia, 115
Chai, Ming, 88
Chikalov, Igor, 12
Czaja, Ludwik, 103
Dutta Soma, 126
Dydo, Łukasz, 115
Fryc, Barbara, 199
Garanina, Natalia, 134
Gora, Paweł, 146
Gori, Roberta, 31
Grabowski, Adam, 158
Gruska, Damas P., 170, 247
Grzes, Tomasz, 235
Hajder, Mirosław, 182
Heitmann, Frank, 76
Janusz, Andrzej, 191
Jaskula, Bolesław, 196
Jaszuk, Marek, 199
Köhler-Bußmeier, Michael, 76
Kolbusz, Janusz, 182, 223
Kopczynski, Maciej, 235
Korostenskiy, Roman, 182
Kováč, Michal, 247
Levi, Francesca, 31
Mikulski, Łukasz, 44
Milazzo, Paolo, 31
Moshkov, Mikhail, 1, 4, 12
Mroczek, Teresa, 199
Nowak-Brzezinska, Agnieszka, 64
Pancerz, Krzysztof, 196
Pardel, Przemysław W., 115, 146
Popova-Zeugmann, Louchka, 3
Przybyła-Kasperek, Małgorzata, 211
Rosiak, Mariusz, 191
Rozycki, Paweł, 223
Schlingloff, Bernd-Holger, 88
Sidorova, Elena, 134
Skowron, Andrzej, 2
Slezak, Dominik, 191
Sokołowska, Barbara, 115
Stawicki, Sebastian, 191
Stepaniuk, Jarosław, 235
Szkola, Jarosław, 196
Tröger, Peter, 22
Wakulicz-Deja, Alicja, 64
Wasilewski Piotr, 126
Werner, Matthias, 22
Zareba, Lech, 115
Zielosko, Beata, 4, 12