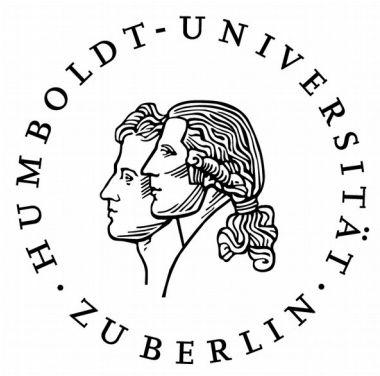
HUMBOLDT-UNIVERSITÄT ZU BERLIN

Concurrency, Specification and Programming
CS&P'2018

Berlin
September 24 - September 26, 2018

edited by Holger Schlingloff & Samira Akili

Informatik-Bericht Nr. 248



INFORMATIK-BERICHTE

Herausgeber: Professoren des Institutes für Informatik
Redaktion: Publikationsstelle, Tel. (+49 30) 2093 3114
Druckerei: Humboldt-Universität zu Berlin
ISSN: 0863 - 095X

Die Reihe Informatik-Berichte erscheint aperiodisch.

Prof. Holger Schlingloff, Samira Akili
Institut für Informatik
Sitz: Rudower Chaussee 25
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin

hs@informatik.hu-berlin.de, akilsami@cms.hu-berlin.de

September 2018

Preface

This volume contains the papers presented at CS&P'18, the 27^{th} International Workshop on Concurrency, Specification and Programming, held on September 24-26 2018 in Berlin.

Since the early seventies Warsaw University and Humboldt University have alternately organized an annual workshop - since the early nineties known as CS&P. Over time, it has grown from a bilateral seminar to a well-known meeting attended also by colleagues from many other countries than Poland and Germany.

During the three-day meeting, there are 8 sessions and a number of short presentations on current and emerging topics, as well as open discussions. While sessions on *Concurrency*, *Verification* and *Programming* have always been part of the CS&P, we are delighted to add sessions on the topics *Learning Systems* and *Reaction Systems* to this years program. Furthermore, there are two invited talks: Prof. Matthias Weidlich will give a presentation on the "Formal Analysis of Complex Event Processing" and Prof. Wojciech Penczek will speak on "Improving Efficiency of Model Checking for Variants of Alternating-time Temporal Logic".

This volume contains 21 papers supplementing the presentations and invited talks, selected from the submissions by the program committee. Following the workshops tradition, we strive to retain an informal working atmosphere. Therefore, the proceedings includes drafts and extended abstracts as well as fully elaborated contributions.

The proceedings are published by Humboldt University and CEUR. The editors would like to thank the university's printing office, the team at CEUR Workshop Proceedings, and EasyChair for their help in producing this publication.

September 26, 2018 Holger Schlingloff & Samira Akili

Table of Contents

Formal Analysis of Complex Event Processing - Potential and Challenges 1
Matthias Weidlich

Opacity-enforcing for Process Algebras 3
Damas Gruska and M. Carmen Ruiz

Spreading information in distributed systems using Gossip algorithm 15
Andrzej Barczak and Michał Barczak

Extensions of Elementary Cause-Effect Structures 27
Ludwik Czaja

Automated Comparative Study of Some Generalized Rough Approximations 39
Adam Grabowski

The Bayes Theorem Counterpart in Mass-Based Rough Mereology 47
Lech Polkowski

On some heuristic method for optimal database workload reconstruction 57
Marcin Zimniak, Marta Burzańska and Bogdan Franczyk

Automated validation of big data classifiers on multiple diverse datasets 69
Przemysław Czaus

Investigating Characteristics and Differences Between Easy and Hard SAT Instances (Extended Abstract) 73
Teofil Sidoruk

Formal Semantics for Probabilistic Verification of Stochastic Regular Expressions 81
Sinem Getir, Esteban Pavese and Lars Grunske

More about left recursion in PEG 91
Roman Redziejowski

Improving Efficiency of Model Checking for Variants of Alternating-time Temporal Logic . 103
Wojciech Penczek

Linking Exploration Systems with Local Logics over Information Systems 107
Andrzej Skowron, Soma Dutta and Grzegorz Rozenberg

Simulating Gene Regulatory Networks using Reaction Systems 119
Roberto Barbuti, Pasquale Bove, Roberta Gori, Francesca Levi and Paolo Milazzo

Hidden States in Reaction Systems 133
Roberta Gori, Damas Gruska and Paolo Milazzo

Preserving Behavior in Transition Systems from Event Structure Models 145
Irina Virbitskaite and Nataliya Gribovskaya

Compositional Expressiveness of Hybrid Automata 159
Jafar Akhundov, Michael Reißner and Matthias Werner

The Influence of the Test Operator on the Expressive Powers of PDL-Like Logics 171
Linh Anh Nguyen

Deep Learning guinea pig image classification using Nvidia DIGITS and GoogLeNet 185
Lukasz Zmudzinski

A Novel Ensemble Model - The Random Granular Reflections 197
Piotr Artiemjew and Krzysztof Ropiak

The Hierarchical Learning Algorithm for Deep Neural Networks 209
Stanisław Płaczek and Aleksander Płaczek

Author and Keyword Index 221
(indexed by article number)

Program Committee

Ludwik Czaja	Institute of Informatics, Warsaw University
Soma Dutta	University of Calcutta
Wojtek Jamroga	Polish Academy of Sciences
Wojciech Penczek	Institute of Computer Science of PAS
Lech Polkowski	Polish-Japanese Institute of Information Technology
Holger Schlingloff	Fraunhofer FOKUS and Humboldt University
Edip Senyurek	Vistula University
Andrzej Skowron	Warsaw University
Zbigniew Suraj	Chair of Computer Science, University of Rzeszów, Poland
Marcin Szczuka	Institute of Informatics, The University of Warsaw
Dmitry Zaitsev	Daze

Additional Reviewers

Akili, Samira
Bazan, Jan
Gburzynski, Pawel
Knapik, Michal
Lorenz, Felix
Meski, Artur
Placzek, Stanislaw
Salwicki, Andrzej
Zaitsev, Dmitry A.
Zbrzezny, Agnieszka

Formal Analysis of Complex Event Processing - Potential and Challenges

Matthias Weidlich

Deparment of Computer Science, Humboldt-Universität zu Berlin
matthias.weidlich@hu-berlin.de

Extended Abstract

Complex event processing (CEP) emerged as a paradigm to build systems that react to *situations of interest* [1]. By evaluating continuous queries over streams of events, CEP systems provide the foundation for re-active and pro-active applications in domains such as healthcare and urban transportation. Models for CEP are an active area of research and various languages for the definition of event stream queries have, so far, been proposed. While each of them provides a different syntax and semantics, they typically adopt point-based event semantics (i.e., the occurrence of an event is atomic), an attribute-based data model (an event carries payload, which is structured as key-value pairs), and comprise a set of common query operators [7], such as *sequencing* of events in terms of their temporal order; *negation* to check for the absence of an event; and *windows* to bound the temporal interval in which events are considered relevant to the query.

Common CEP applications face high-velocity event streams, which renders the evaluation of queries a performance bottleneck. Therefore, various algorithms and architectures for efficient CEP have been proposed in recent years, including techniques for parallelisation and distribution of query evaluation [3], semantic query rewriting [6], or sub-pattern sharing [4]. Moreover, reflecting on the challenges induced by distributed event sources, techniques to achieve robustness of CEP against out-of-order arrivals of events have been developed [2].

Despite all these advancements, we argue that most of these techniques adopt a pragmatic view—they strive for a technical solution of the issues as they emerge in a specific application. We therefore advocate the design of formal methods to guide the design and implementation of CEP applications. Specifically, we suggest to rely on well-established formalisms for concurrent systems to reason about the following aspects of CEP applications:

- *Query verification:* Formal analysis may reveal whether a query deployed in a CEP system can match at all, specifically if event streams satisfy domain-specific constraints.
- *Sound parallelisation:* Formal analysis may identify non-determinism in query evaluation that is introduced through paralelisation schemes for queries that potentially interact with each other.
- *Robustness guarantees:* Formal analysis may enable conclusions on the errors introduced by out-of-order event arrivals at a CEP system, thereby giving robustness guarantees.

In the light of the above reasoning tasks, we developed a formal model of CEP applications that is grounded in Petri-nets [5]. The choice of this formalism is motivated by their concurrent and local semantics, along with the broad availability of analysis algorithms and tool support.

Major challenges in this endeavour, however, have been the integration of the various perspectives of a CEP application in a single model: It requires a comprehensive formalisation of the semantics of event streams, event queries, and evaluation architectures, properly capturing their interplay. For instance, it is not sufficient to simply capture that an event occurred, but several modalities have to be encoded on the formal level: An event may have occurred, but may be consumed by a match of an event query and, thus, no longer be available to construct further matches. In such a case, the interplay with common evaluation architectures needs to be taken into account. If an event can be consumed only by a single match of a query, sequential evaluation of various match candidates of a query may lead to a different result compared to their concurrent evaluation.

Using our formal model, we are able to approach reasoning on the aforementioned aspects of CEP applications through standard reachability analysis. The question of whether a specific query can match translates into the common problem of identifying whether a specific state (i.e., a marking in the Petri-net) can be obtained from the initial state. Based thereon, conclusions can be drawn on the general possibility of generating matches; on the changes in the sets of matches obtained under different parallelisation schemes; and on the implications of out-of-order arrivals of events, whether they potentially lead to false positives and false negatives in query evaluation.

References

1. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. **44**(3), 15:1–15:62 (2012).
2. Liu, M., Li, M., Golovnya, D., Rundensteiner, E.A., Claypool, K.T.: Sequence pattern query processing over out-of-order event streams. In: ICDE. IEEE (2009).
3. Mayer, R., Slo, A., Tariq, M.A., Rothermel, K., Gräber, M., Ramachandran, U.: SPECTRE: supporting consumption policies in window-based parallel complex event processing. In: Middleware. pp. 161–173. ACM (2017).
4. Ray, M., Lei, C., Rundensteiner, E.A.: Scalable pattern sharing on event streams. In: SIGMOD. pp. 495–510. ACM (2016).
5. Reisig, W.: Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies. Springer (2013).
6. Weidlich, M., Ziekow, H., Gal, A., Mendling, J., Weske, M.: Optimizing event pattern matching using business process models. IEEE Trans. Knowl. Data Eng. **26**(11), 2759–2773 (2014).
7. Zhang, H., Diao, Y., Immerman, N.: On complexity and optimization of expensive queries in complex event processing. In: SIGMOD. pp. 217–228. ACM (2014).

Opacity-enforcing for Process Algebras *

Damas P. Gruska[1] and M. Carmen Ruiz[2]

[1] Comenius University, Slovakia
[2] Universidad de Castilla-La Mancha, Spain

Abstract. Supervisory control as a way how to guarantee security of processes is discussed and studied. We work with a security property called processes opacity and we investigate how it can be enforced. Supervisors can restrict behaviour of the original systems by enabling or disabling some actions to guarantee its security. We study maximal supervisors as the least restricting supervisory control processes. Moreover, we study also enhanced supervisory control which can add idling between system's action to prevent timing attacks.

Keywords: security, opacity, process algebras, information flow, supervisory control

1 Introduction

The great revolution brought about by the internet of things involves the emergence of new devices, new protocols and, of course, new security needs to fulfill the new requirements. New protocols come into operation before they have been evaluated in depth. This leads to the appearance of new versions of the protocol that is not always compatible with its predecessors and that companies will not always incorporate in their devices with sufficient speed. In addition, these solutions usually require downloading a new code and this itself is open to security attacks. This lack of security has been detected even in our own works. For example in [Gar16] we present an architecture for Wireless Sensor and Actuator Networks (WSAN) using the Bluetooth Low Energy (BLE) and TCP/IP protocols in conjunction, which make necessary to include bridges that lack basic security requirements. Another example can be found in [Hor17] where we propose a new packet format and a new BLE mesh topology, with two different configurations: Individual Mesh and Collaborative Mesh. All these represent our motivation to study applicability of formal models and formal methods to define and enforce system's security. As regards formalism, we will work with timed process algebra. Then we exploit information flow based security properties (see [GM82]) which 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 or states can be deduced. This approach has found many reformulations and among them opacity (see [BKR04,BKMR06]) could be considered as the

* Work supported by the grant VEGA 1/0778/18.

most general one and many other security properties could be viewed as its special cases (see, for example, [Gru15,Gru12,Gru11,Gru10,Gru08,Gru07]). Opacity properties could be divided into two types: language based opacity, expressing security (privacy) of system's actions or traces of actions and state based one, concentrating on system's states (see an overview paper [JLF16]). The former one is much more studied for process algebra's formalism. But also for the later one there is some research already done. In [Gru15] we consider an intruder who wants to discover whether a process reaches a confident state. Resulting security property is called process opacity. It turned out that in this way some new security flaws could be expressed. If a process is not secure with respect to process opacity we can either re-design its behavior, what might be costly, difficult or even impossible, in the case that it is already part of a hardware solution, proprietary firmware and so on or we can use supervisory control (see [RW89]) to restrict system's behaviour in such a way that the system becomes secure. A supervisor can see (some) system's actions and can control (disable or enable) some set of system's action. In this way it restricts system's behaviour to guarantee its security. This is a trade-off between security and functionality. But in many cases it is not a fatal problem. Suppose that a communication protocol can reach (with a low probability) a state which is not secure. In that case the transmission of a packet is interrupted and it should start from the begging. Sometimes this restriction has even smaller impact on system's behavior. Suppose that the system can perform action a and b in an arbitrary order but only a sequence $b.a$ could leak some classified information about intermediate states. Restricting this sequence make system secure but could not have influence on overall system's functionality. In this paper we do not assume any relation among a set of actions visible for an intruder, a set of actions visible for a controller and a set of controllable actions, i.e. sets E_I, E_S, E_C, respectively, similarly to [TLSG18]. Note that in [DDM10] it is assumed that $E_I \subseteq E_S$ (or $E_S \subseteq E_I$) and $E_C \subseteq E_S$. In [YL10] $E_I \subseteq E_S$ is assumed and in [TLSG16] $E_C \subseteq E_S$ is assumed. As regards the related work, besides already mentioned works there is a large body of work on controller synthesis in temporal model checking. From the rest we mention just two papers. In [RS01] the idea of controller to enforce secure information flow is discussed for language based security in process algebra setting. Opacity-enforcing (called strategic noninterference) was proposed and investigated for transition systems in [JT15].

Timing attacks, as side channel attacks, represent serious threat for many systems. They allow intruders "break" "unbreakable" systems, algorithms, protocols, etc. Even relatively recently discovered possible attacks on most of currently used processors (Meltdown and Spectre) also belong to timing attacks. To protect systems against some type of timing attacks we propose to enhance capabilities of the supervisory control. Such controller can add some idling between actions to enforce process's security.

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 present supervisory control. The next section contains some basic definition on informa-

tion flow security and process opacity. Sections 5 and 6 deals with supervisory and enhanced supervisory control for process opacity, respectively.

2 Timed Process Algebra

In this section we define Timed Process Algebra, TPA for short. TPA is based on Milner's CCS but the special time action t which expresses elapsing of (discrete) time is added. The presented language is a slight simplification of Timed Security Process Algebra introduced in [FGM00]. We omit an explicit idling operator ι 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" (i.e. for example, the process $a.Nil$ can perform t action since it is not contained the process specification). But in both cases internal communications have priority to action t in the parallel composition. 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 determinancy* 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 τ and t, and such that for every $a \in A$ there exists $\overline{a} \in A$ and $\overline{\overline{a}} = a$. We define $Act = A \cup \{\tau\}, At = A \cup \{t\}, Actt = Act \cup \{t\}$. We assume that $a, b, \ldots$ range over A, $u, v, \ldots$ range over Act, and $x, y \ldots$ range over $Actt$. Assume the signature $\Sigma = \bigcup_{n\in\{0,1,2\}} \Sigma_n$, where

$$\begin{aligned}
\Sigma_0 &= \{Nil\} \\
\Sigma_1 &= \{x. \mid x \in A \cup \{t\}\} \cup \{[S] \mid S \text{ is a relabeling function}\} \\
&\quad \cup \{\backslash M \mid M \subseteq A\} \\
\Sigma_2 &= \{|, +\}
\end{aligned}$$

with the agreement to write unary action operators in prefix form, the unary operators $[S], \backslash M$ in postfix form, and the rest of operators in infix form. Relabeling functions, $S : Actt \to Actt$ are such that $\overline{S(a)} = S(\bar{a})$ for $a \in A, S(\tau) = \tau$ and $S(t) = t$.

The set of TPA terms over the signature Σ 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 an usual definition of opened and closed terms where μX is the only binding operator. Closed terms which are t-guarded (each occurrence of X is within some subterm $t.A$ i.e. between any two t actions only finitely many non timed actions can be performed) are called TPA processes.

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 $Actt$. The transition relation $\rightarrow$ is a subset of TPA $\times$ $Actt$ $\times$ TPA. We write $P \xrightarrow{x} P'$ instead of $(P, x, P') \in \rightarrow$ and $P \xrightarrow{x} \not\rightarrow$ 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 plus the following inference rules:

$$\frac{}{Nil \xrightarrow{t} Nil} \quad A1 \qquad \frac{}{u.P \xrightarrow{t} u.P} \quad A2$$

$$\frac{P \xrightarrow{t} P', Q \xrightarrow{t} Q', P \mid Q \xrightarrow{\tau} \not\rightarrow}{P \mid Q \xrightarrow{t} P' \mid Q'} \; Pa \qquad \frac{P \xrightarrow{t} P', Q \xrightarrow{t} Q'}{P + Q \xrightarrow{t} P' + Q'} \; S$$

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 (Pa). A run of time is deterministic (S) i.e. performing of t action does not lead to the choice between summands of $+$. In the definition of the labeled transition system we have used negative premises (see Pa). In general this may lead to problems, for example with consistency of the defined system. We avoid these dangers by making derivations of τ independent of derivations of t. For an explanation and details see [Gro90].

For $s = x_1.x_2.\ldots.x_n, x_i \in Actt$ we write $P \xrightarrow{s}$ instead of $P \xrightarrow{x_1} \xrightarrow{x_2} \ldots \xrightarrow{x_n}$ and we say that s is a trace of P. The set of all traces of P will be denoted by $Tr(P)$. By ϵ we will denote the empty sequence of actions, by $Succ(P)$ we will denote the set of all successors of P i.e. $Succ(P) = \{P' | P \xrightarrow{s} P', s \in Actt^*\}$. If the set $Succ(P)$ is finite we say that P is a finite state process. We define modified transitions $\stackrel{x}{\Rightarrow}_M$ which "hide" actions from M. Formally, we will write $P \stackrel{x}{\Rightarrow}_M P'$ for $M \subseteq Actt$ iff $P \xrightarrow{s_1} \xrightarrow{x} \xrightarrow{s_2} P'$ for $s_1, s_2 \in M^\star$ and $P \stackrel{s}{\Rightarrow}_M$ instead of $P \stackrel{x_1}{\Rightarrow}_M \stackrel{x_2}{\Rightarrow}_M \ldots \stackrel{x_n}{\Rightarrow}_M$. We will write $P \stackrel{x}{\Rightarrow}_M$ if there exists P' such that $P \stackrel{x}{\Rightarrow}_M P'$. We will write $P \stackrel{x}{\widehat{\Rightarrow}}_M P'$ instead of $P \stackrel{\epsilon}{\Rightarrow}_M P'$ if $x \in M$. Note that $\stackrel{x}{\Rightarrow}_M$ is defined for arbitrary action x but in definitions of security properties we will use it for actions (or sequence of actions) not belonging to M. We can extend the definition of $\Rightarrow_M$ for sequences of actions similarly to $\xrightarrow{s}$. Let $s \in Actt^\star$. By $|s|$ we will denote the length of s i.e. a number of action contained in s. By $s|_B$ we will denote the sequence obtained from s by removing all actions not belonging to B. For example, $|s|_{\{t\}}|$ denote a number of occurrences of t in s, i.e. time length of s. By $Sort(P)$ we will denote the set of actions from A which can be performed by P. The set of traces of process P is defined as $L(P) = \{s \in Actt^\star | \exists P'.P \stackrel{s}{\Rightarrow} P'\}$. The set of weak timed traces of process P is defined as $L_w(P) = \{s \in (A \cup \{t\})^\star | \exists P'.P \stackrel{s}{\Rightarrow}_{\{\tau\}} P'\}$. Two processes P and Q are weakly timed trace equivalent ($P \simeq_w Q$) iff $L_w(P) = L_w(Q)$. We conclude this section with definitions of M-bisimulation and weak timed trace equivalence.

Definition 1. *Let $(TPA, Actt, \rightarrow)$ be a labelled transition system (LTS). A relation $\Re \subseteq TPA \times TPA$ is called a* M-bisimulation *if it is symmetric and it satisfies the following condition: if $(P,Q) \in \Re$ and $P \xrightarrow{x} P', x \in Actt$ then there exists a process Q' such that $Q \stackrel{\widehat{x}}{\Rightarrow}_M Q'$ and $(P',Q') \in \Re$. Two processes P, Q are M-bisimilar, abbreviated $P \approx_M Q$, if there exists a M-bisimulation relating P and Q.*

3 Supervisory control

In this section we introduce some basic concepts of supervisory control theory. For more details see [RW89]. Let us assume deterministic finite automaton (DFA) $G = (X, E, \delta, x_0)$, where X is the finite set of states, E is the set of events, $\delta : X \times E \rightarrow X$ is the (partial) transition function, $x_0 \in X$ is the initial state. The transition function can be naturally extended to strings of events. The generated language of $G = (X, E, \delta, x_0)$ is defined as $L(G) = \{s, s \in E^*$ such that $\delta(x_0, s)$ is defined$\}$.

The goal of supervisory control is to design a control agent (called supervisor) that restricts the behavior of the system within a specification language $K \subseteq L(G)$. The supervisor observes a set of observable events $E_S \subseteq E$ and is able to control a set of controllable events $E_C \subseteq E$. The supervisor enables or disables controllable events. When an event is enabled (resp., disabled) by the supervisor, all transitions labeled by the event are allowed to occur (resp., prevented from occurring). After the supervisor observes a string generated by the system it tells the system the set of events that are enabled next to ensure that the system will not violate the specification.

A supervisor can be represented by $Sup = (Y, E_S, \delta_s, y_0, \Psi)$, where (Y, E_S, δ_s, y_0) is an automaton and $\Psi : Y \rightarrow \{E' \subseteq E | E_{UC} \subseteq E'\}$ where $E_{UC} = E \setminus E_C$ specifies the set of events enabled by the supervisor in each state. System G under the control of a suitable supervisor Sup is denoted as Sup/G, and it satisfies $L(Sup/G) \subseteq K$.

Definition 2 (Controllability). *Given a $DFAG$, a set of controllable events E_C, and a language $K \subseteq L(G)$, K is said to be controllable (wrt $L(G)$ and E_C) if*

$$\bar{K} E_{UC} \cap L(G) \subset \bar{K}$$

where $\bar{K}$ is the prefix closer of K.

The controllability of K requires that for any prefix $s, s \in K$, if s followed by an uncontrollable event $e \in E_{UC}$ is in $L(G)$, then it must also be a prefix of a string in K.

Definition 3 (Observability). *Given a $DFAG$, a set of controllable events E_C , a set of observable events E_S , and a language $K \subseteq L(G)$, K is said to be observable (wrt $L(G)$, E_S and E_C) if for all $s, s' \in \bar{K}$ and all $e \in E_C$ such that $se \in L(G)$, $s =_S s'$ ($s =_S s'$ is means that strings are equal with respect to the set E_S), $s.e \in \bar{K}$.*

Observability requires that supervisors observation of the system (i.e., the projection of s on E_S) provides sufficient information to decide after the occurrence of a controllable event whether the resultant string is still in $\bar{K}$.

Proposition 1. *Let $K \subseteq L(G)$ be a prefix-closed nonempty language, E_C the set of controllable events and E_S the set of observable events. There exists a supervisor Sup such that $L(Sup/G) = K$ if and only if K is controllable and observable.*

4 Information flow

In this section we will present our working security concept. First we define the absence-of-information-flow property - Strong Nondeterministic Non-Interference (SNNI, for short, see [FGM00]). Suppose that all actions are divided into two groups, namely public (low level) actions L and private (high level) actions H. It is assumed that $L \cup H = A$. SNNI property assumes an intruder who tries to learn whether a private action was performed by a given process while (s)he can observe only public ones. If this cannot be done then the process has SNNI property. Namely, process P has SNNI property (we will write $P \in SNNI$) if $P \setminus H$ behaves like P for which all high level actions are hidden (namely, replaced by action τ) for an observer. To express this hiding we introduce the hiding operator $P/M, M \subseteq A$, for which it holds that 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 $a \in M \cup \overline{M}$. Formally, we say that P has SNNI property, and we write $P \in SNNI$ iff $P \setminus H \simeq_w P/H$. A generalization of this concept is given by opacity (this concept was exploited in [BKR04], [BKMR06] and [Gru07] in a framework of Petri Nets, transition systems and process algebras, respectively). Actions are not divided into public and private ones at the system description level but a more general concept of observations and predicates are exploited. A predicate is opaque if for any trace of a system for which it holds, there exists another trace for which it does not hold and the both traces are indistinguishable for an observer (which is expressed by an observation function). This means that the observer (intruder) cannot say whether a trace for which the predicate holds has been performed or not. Now let us assume a different scenario, namely that an intruder is not interested in traces and their properties but he or she tries to discover whether a given process always reaches a state with some given property which is expressed by a (total) predicate. This property might be process deadlock, capability to execute only traces with time length less then n time unites, capability to perform at the same time actions form a given set, incapacity to idle (to perform t action) etc. We do not put any restriction on such predicates but we only assume that they are consistent with some suitable behavioral equivalence. The formal definition follows.

Definition 4. *We say that the predicate ϕ over processes is consistent with respect to relation $\cong$ if whenever $P \cong P'$ then $\phi(P) \Leftrightarrow \phi(P')$.*

As the consistency relation $\cong$ we could take bisimulation ($\approx_\emptyset$), weak bisimulation ($\approx_{\{\tau\}}$) or any other suitable equivalence. A special class of such predicates are such ones (denoted as $\phi^Q_{\cong}$) which are defined by a given process Q and equivalence relation $\cong$ i.e. $\phi^Q_{\cong}(P)$ holds iff $P \cong Q$.

We suppose that the intruder can observe only some activities performed by the process. Hence we suppose that there is a set of public actions which can be observed and a set of hidden (not necessarily private) actions. To model such observations we exploit the relation $\stackrel{s}{\Rightarrow}_M$ where actions from M are those ones which could not be seen by the observer. The formal definition of process opacity (see [Gru15]) is the following.

Definition 5 (Process Opacity). *Given process P, a predicate ϕ over processes is process opaque w.r.t. the set M if whenever $P \stackrel{s}{\Rightarrow}_M P'$ for $s \in (Actt \setminus M)^*$ and $\phi(P')$ holds then there exists P'' such that $P \stackrel{s}{\Rightarrow}_M P''$ and $\neg\phi(P'')$ holds. The set of processes for which the predicate ϕ is process opaque w.r.t. to the M will be denoted by POp^ϕ_M.*

Note that if $P \cong P'$ then $P \in POp^\phi_M \Leftrightarrow P' \in POp^\phi_M$ whenever ϕ is consistent with respect to $\cong$ and $\cong$ is such that it is a subset of the trace equivalence (defined as $\simeq_w$ but instead of $\stackrel{s}{\Rightarrow}_{\{\tau\}}$ we use $\stackrel{s}{\Rightarrow}_\emptyset$).

$$P \quad \stackrel{s}{\Longrightarrow}_M \quad \phi(P')$$

$$P \quad \stackrel{s}{\Longrightarrow}_M \quad \neg\phi(P'')$$

Fig. 1. Process opacity

5 Supervisory Control of Process Opacity

In this section we will concentrate on enforcing process opacity, namely, how to guarantee that there is no leakage of information on validity of ϕ in a current state, i.e. security with respect to process opacity. Let $M \subseteq Actt$ by $\bar{M}$ we will denote the complement of M i.e. $\bar{M} = Actt \setminus M$. Let $s \in Actt^*$, by $s_{\bar{M}}$ we denote the string obtained form s by removing all elements belonging to M. Formally, $\epsilon_{\bar{M}} = \epsilon$, $s.x_{\bar{M}} = s.x$ iff $x \notin M$ and $s.x_{\bar{M}} = s$ iff $x \in M$. We can extend this definition to a set of strings. Let $T \subseteq Actt^*$ then $T_{\bar{M}} = \{s_{\bar{M}} | s \in T\}$.

Now let as suppose that process P is not secure with respect to process opacity POp^ϕ_M i.e. $P \notin POp^\phi_M$. That means that there exists $s \in L(P)_{\bar{M}}$ such that $P \stackrel{s}{\Rightarrow}_M P'$ and $\phi(P')$ holds then there does not exist P'' such that $P \stackrel{s}{\Rightarrow}_M P''$ and $\neg\phi(P'')$ holds. Hence, by observing s, an intruder knows that a state satisfying ϕ has been reached. For security reasons we want to prohibit such computations what will be the role for the supervisory control. Formally, let $K, K \subseteq L(P)_{\bar{M}}$ is a set of safe observations, i.e. for every $s \in K$, $P \stackrel{s}{\Rightarrow}_M P'$ and $\phi(P')$ does not hold or if it holds then there exists P'' such that $P \stackrel{s}{\Rightarrow}_M P''$ and

$\neg\phi(P'')$ holds. Clearly, if $P \in POp^{\phi}_{M}$ then $K = L(P)_{\bar{M}}$, otherwise $K \subset L(P)_{\bar{M}}$ but $K \neq L(P)_{\bar{M}}$. The aim of the control is to design a supervisor Sup which will restrict behaviour of the original process P in such a way that for the resulting process Sup/P we have $L(Sup/P) \subseteq K$. Note that we do not assume any relations among set of actions visible for an intruder, a set of actions visible for a controller and a set of controllable actions, i.e. sets $E_I (E_I = \bar{M}), E_S, E_C$, respectively, similarly to [TLSG18]. Note that in [DDM10] it is assumed that $E_I \subseteq E_S$ (or $E_S \subseteq E_I$) and $E_C \subseteq E_S$. In [YL10] $E_I \subseteq E_S$ is assumed and in [TLSG16] $E_C \subseteq E_S$ is assumed.

Example 1. Let $P = c.(a.b.Nil + b.(a.Nil + d.Nil))$, $M = \{c, d\}$ and predicate ϕ is defined as follows: $\phi(Q)$ holds iff $Q \xrightarrow{d}$. Then it is easy to check that $P \notin POp^{\phi}_{M}$. The execution of $c.b$ (visible as b) at the beginning leads to the state satisfying ϕ but no execution visible as b can lead to a state not satisfying ϕ.

Now we will model supervisory control by means a special process Sup. Process Sup runs in parallel with P, communicates with environment via actions $Sort(P)$ and internally with P by actions renamed by function f which maps every action a from $Sort(P)$ to a new "ghost" action a' (see Fig. 2). The formal definition of process supervisor is the following.

Fig. 2. Supervisory Control

Definition 6 (Process Supervisor). *Given process P, a process Sup is called supervisor if $L_w(Sup/P) \subseteq L_w(P)$ where $Sup/P = (P[f]|Sup) \setminus f(Sort(P))$ where $f : Sort(P) \to Sort(P)'$ where $Sort(P)' = \{x' | x \in Sort(P), x \neq \tau\}$.*

We will use a process supervisor to restrict behaviour of the original process in such a way that the resulting process becomes secure with respect to process opacity.

Definition 7 (Process Supervisor for Process Opacity). *Given process P, process Sup is called supervisor for opacity property POp^{ϕ}_{M} iff $P \notin POp^{\phi}_{M}$ but $Sup/P \in POp^{\phi}_{M}$. By $Sup(P, POp^{\phi}_{M})$ we will denote the set of all supervisors for opacity property POp^{ϕ}_{M} for a given process P.*

Example 2. Let us continue with the Example 1. Let $Sup_1 = c'.c.Nil$ $Sup_2 = c'.c.a'.aNil$ and $Sup_3 = c'.c.a'.a.b'.b.Nil$ then it is easy to check that all processes Sup_i are process supervisors for P and opacity property POp^{ϕ}_{M}. Actually these process supervisors restrict the execution of action b immediately after c.

Clearly, $Sup(P, POp^{\phi}_{M}) \neq \emptyset$ since $Nil \in Sup(P, POp^{\phi}_{M})$. Note that supervisor Nil restricts all behaviour of P which consequently becomes trivially secure. We can formulate some properties of the set $Sup(P, POp^{\phi}_{M})$.

Proposition 2. *Let* $Sup_1, Sup_2 \in Sup(P, POp^{\phi}_M)$. *Then* $Sup_1 + Sup_2 \in Sup(P, POp^{\phi}_M)$.

Proof. The main idea. The first actions which is performed by $(Sup_1 + Sup_2)/P$ is performed either by Sup_1 or by Sup_2.

Proposition 3. *Let* $Sup_1 \in Sup(P, POp^{\phi}_M)$ *and* $Sup_1 \approx_{\emptyset} Sup_2$. *Then* $Sup_2 \in Sup(P, POp^{\phi}_M)$.

Proof. Sketch. The proof follows from the the fact that trace equivalence is congruence i.e. for $Sup_1 \approx_{\emptyset} Sup_2$ we have $(P[f]||Sup_1)\backslash f(Sort(P)) \approx_{\emptyset} (P[f]||Sup_1)\backslash f(Sort(P))$ and so $L((P[f]||Sup_1) \setminus f(Sort(P))) = L((P[f]||Sup_2) \setminus f(Sort(P)))$ i.e.$L(Sup_1/P) = L(Sup_2/P)$.

To guarantee a minimal restriction of process behaviour our aim is to find a maximal process supervisor in a sense that it minimally restricts behavior of the original process. The formal definition is the following.

Definition 8 (Maximal Process Supervisor for Process Opacity). *Process* $Sup \in Sup(P, POp^{\phi}_M)$ *is called maximal process supervisor for process opacity* POp^{ϕ}_M *iff for every* $Sup' \in Sup(P, POp^{\phi}_M)$ $L(Sup'/P) \subseteq L(Sup/P)$.

Example 3. Let us continue with the Examples 1 and 2. It is easy to check that process Sup_3 is a maximal process supervisor for P and opacity property POp^{ϕ}_M. Processes Sup_1 and Sup_2 are not maximal process supervisors for P and opacity property POp^{ϕ}_M.

Unfortunately it is undecidable to verify whether a process Sup is a process supervisors for P and opacity property POp^{ϕ}_M as it is stated by the following proposition.

Proposition 4. *The property that Sup is a process supervisor for process opacity for process P is undecidable in general.*

Proof. The proof is based on an idea that already process opacity is undecidable (see Proposition 2. in [Gru15]). Suppose that the property is decidable. Let $Sup = \mu X. \sum_{x \in Actt} x'.x.X$ i.e Sup does not restrict anything. We have that Sup is a process supervisor for process opacity for process P iff $P \in POp^{\phi}_M$. Hence we would be able to decide process opacity what contradicts its undecidability.

Corollary. The property that Sup is a maximal process supervisor for process opacity for process P is undecidable in general.

To obtain a decidable variant of the previous property we put some restriction on process predicates. First we model predicates by special processes called tests. For now we assume that action τ is not visible for an intruder, i.e. $\tau \in M$. The tests communicate with processes and produce $\surd$ action if corresponding predicates hold for the processes. In the subsequent proposition we show how to exploit this idea to express process opacity by means of appropriate M-bisimulation.

Definition 9. *We say that the process T_ϕ is the test representing predicate ϕ if $\phi(P)$ holds iff $(P|T_\phi)\setminus At \approx_t \surd.Nil$ where $\surd$ is a new action indicating a passing of the test. If T_ϕ is the finite state process we say that ϕ is the finitely definable predicate.*

Suppose that both ϕ and $\neg\phi$ are the finitely definable predicates. Then we can reduce checking whether *Sup* is a process supervisor for process opacity to checking bisimulation (see Proposition 4. in [Gru15]). Since we can reduce the problem of decidability to finite automata (see [TLSG18]) we obtain the following result.

Proposition 5. *Let ϕ and $\neg\phi$ are finitely definable predicates. The property that Sup is a process supervisor for process opacity for finite state process P is decidable. Moreover, we can always find a maximal supervisor for process opacity.*

6 Enhanced Supervisory Control

Time attacks belong to powerful tools for attackers who can observe or interfere with systems in real time. By the presented formalism we can distinguish timing attacks. Suppose that $P \notin POp^{\phi}_{M}$ but $P \in POp^{\phi}_{M\cup\{t\}}$. This means that an attack is possible only for an observer who can see elapsing of time, i.e. there is a possibility of timing attacks. To prevent them, we can use process supervisor which restricts process behaviour with respect to actions from A or we introduce a new type of process supervisor, called *enhanced process supervisor* which can add some idling between actions to ensure that the resulting process becomes secure with respect to timing attacks. In this case the restriction with respect to atomic actions from A could be smaller as in the case of original supervisory control.

Definition 10 (Enhanced Process Supervisor). *Given process P, a process $ESup$ is called enhanced supervisor if whenever $s \in L_w(P)$ then $ESup/P \stackrel{s}{\Rightarrow}_{t,\tau}$ where $ESup/P = (P[f]||ESup) \setminus f(Sort(P))$ where $f : Sort(P) \to Sort(P)'$ where $Sort(P)' = \{x'|x \in Sort(P), x \neq \tau\}$.*

Definition 11 (Enhanced Process Supervisor for Process Opacity). *Given process P, $P \in POp^{\phi}_{M\cup\{t\}}$, a process $ESup$ is called enhanced supervisor for opacity property POp^{ϕ}_{M} iff $P \notin POp^{\phi}_{M}$, but $ESup/P \in POp^{\phi}_{M}$. By $ESup(P, POp^{\phi}_{M})$ we will denote the set of all supervisors for opacity property $EPOp^{\phi}_{M}$ for a given process P.*

Now we can formulate a condition which guarantees existence of an enhanced supervisory control.

Proposition 6. *Let $P \notin POp^{\phi}_{M}$ and there exists P', $P \approx_{\tau,t} P'$ such that $P' \in POp^{\phi}_{M}$. Then there exists nontrivial (not equivalent to Nil with respect to $\approx_\tau$) enhanced supervisory control for P.*

Proof. The main idea. According to the assumption processes P and P' behave essentially in the same way but the later performs more idling between actions from Act. The enhanced supervisory control adds this idling to behaviour of P in such a way that the resulting process is process opaque.

Moreover, the previous proposition has the following consequence which guaranties that the maximal enhanced supervisory control does not restrict action from A.

Corollary. Let $P \in POp^{\phi}_{M\cup\{t\}}$ and $P \notin POp^{\phi}_{M}$. For the maximal enhanced supervisory control for P we have $L(ESup/P)_{\bar{A}} = K_{\bar{A}}$.

7 Conclusions

We have presented the new concepts called supervisory and enhanced supervisory controller for process algebra which enforce the security property called process opacity. Particularly, we have investigated finite state systems and time sensitive observations. A supervisor can see (some) system's actions and can control (disable or enable) some set of system's action. In this way it restricts system's behaviour to guarantee its security. Sometimes either we simply cannot redesign original insecure system which could have, for example, hardware implementation or some small restriction of system's behaviour is not essential for overall system functionality. In the case of enhanced supervisory controller it can only add some idling between actions which would have no influence on system non-timing properties. The presented approach allows us to exploit also process algebras enriched by operators expressing other "parameters" (space, distribution, networking architecture, processor or power consumption and so on). In this way also other types of attacks, which exploit information flow through various covert channels, can be described and enforced. Hence we could obtain security properties which have not only theoretical but also practical value, since many protocols, particularly low level protocols for IoT, could be described by means of some process algebra formalism.

References

[BKR04] Bryans J., M. Koutny and P. Ryan: Modelling non-deducibility using Petri Nets. Proc. of the 2nd International Workshop on Security Issues with Petri Nets and other Computational Models, 2004.

[BKMR06] Bryans J., M. Koutny, L. Mazare and P. Ryan: Opacity Generalised to Transition Systems. In Proceedings of the Formal Aspects in Security and Trust, LNCS 3866, Springer, Berlin, 2006.

[DDM10] Dubreil J., P. Darondeau and H. Marchand: Supervisory control for opacity. IEEE Trans Autom Control 55(5), 2010.

[FGM00] Focardi, R., R. Gorrieri, and F. Martinelli: Information flow analysis in a discrete-time process algebra. Proc. 13^{th} Computer Security Foundation Workshop, IEEE Computer Society Press, 2000.

[Gar16] Garrido C., V. Lopez, T. Olivares and M. C. Ruiz: Architecture Proposal for Heterogeneous, BLE-Based Sensor and Actuator Networks for Easy Management of Smart Homes. 15th ACMIEEE International Conference on Information Processing in Sensor Networks (IPSN), 2016.

[GM04] Gorrieri R. and F. Martinelli: A simple framework for real-time cryptographic protocol analysis with compositional proof rules. Science of Computer Programming, Volume 50, Issues 13, 2004.

[GM82] Goguen J.A. and J. Meseguer: Security Policies and Security Models. Proc. of IEEE Symposium on Security and Privacy, 1982.

[Gro90] Groote, J. F.: Transition Systems Specification with Negative Premises. Proc. of CONCUR'90, Springer Verlag, Berlin, LNCS 458, 1990.

[Gru15] Gruska D.P.: Process Opacity for Timed Process Algebra. In Perspectives of System Informatics, LNCS 8974, 2015.

[Gru12] Gruska D.P.: Informational analysis of security and integrity. Fundamenta Informaticae, vol. 120, Numbers 3-4, 2012.

[Gru11] Gruska D.P.: Gained and Excluded Private Actions by Process Observations. Fundamenta Informaticae, Vol. 109, No. 3, 2011.

[Gru10] Gruska D.P.: Process Algebra Contexts and Security Properties. Fundamenta Informaticae, vol. 102, Number 1, 2010.

[Gru08] Gruska D.P.: Probabilistic Information Flow Security. Fundamenta Informaticae, vol. 85, Numbers 1-4, 2008.

[Gru07] Gruska D.P.: Observation Based System Security. Fundamenta Informaticae, vol. 79, Numbers 3-4, 2007.

[Hor17] Hortelano, D., T Olivares, M.C. Ruiz, M. Carmen, C. Garrido-Hidalgo and V. López: From Sensor Networks to Internet of Things. Bluetooth Low Energy, a Standard for This Evolution. Sensors, vol 17, 2017.

[JLF16] Jacob, R., J.-J.Lesage and J.-M. Faure: Overview of discrete event systems opacity: Models, validation, and quantification, Annual Reviews in Control Volume 41, 2016.

[JT15] Jamroga W. and M. Tabatabaei: Strategic Noninterference. ICT Systems Security and Privacy Protection, SEC 2015.

[RW89] Ramadge P.J.G, Wonham W.M.: The control of discrete event systems. Proc IEEE 77(1):8198, 1989.

[RS01] Ryan P. Y. A. and S. A. Schneider: Process Algebra and Non-Interference. Journal of Computer Security 9(1/2), 2001.

[TLSG18] Tong, Y, Z. Li, Zhiwu, C. Seatzu and A. Giua: Current-state opacity enforcement in discrete event systems under incomparable observations. Discrete Event Dynamic Systems, vol. 28, 2018.

[TLSG16] Tong, Y, Z. Li, Zhiwu, C. Seatzu and A. Giua: Supervisory enforcement of current-state opacity with uncomparable observations. In: Proceedings of the 13th International workshop on discrete event systems, 2016.

[YL10] Yin X. and S. Lafortune: A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems. In: Proceedings of the 2015 American control conference. IEEE, Chicago, 2015.

Spreading information in distributed systems using Gossip algorithm

Andrzej Barczak

Siedlce University, Faculty of Science, Institute of Computer Science,

3 Maja 54, 08-110 Siedlce, Poland

andrzej.barczak@uph.edu.pl

Michal Barczak

Mettler Toledo

ul. Poleczki 21, 02-822 Warsaw, Poland

Abstract. In a following article problem of a information sharing in distributed system is described. Ways of solving that problem with emphasize on Gossip protocol are as well presented. The main goal of the article is to examine and analyze the operation of the Gossip algorithm, taking into consideration chosen mode and values of parameters. At the beginning, problem of exchanging information in distributed system is presented and different algorithms solving it were shown. Afterward, the Gossip algorithm is presented. The principle of its operation, its working modes and models in which the Gossip algorithm can work are described. The pseudo code of the Gossip algorithm in the both SI and SIR model is shown. As well the problem of getting known by the nodes, the topology of the system in which the message is sent, is described. The limitations of the Gossip algorithm are also presented. As part of the article, an application in C# language was made, allowing to examine the Gossip protocol in a laboratory environment. This protocol is highly parametrized and can operate in several modes. The laboratory environment contained minimum six and a maximum ten nodes. The number of iterations needed to achieve consistency for six, eight and ten connected computers was examined. The nodes were connected to each other in topologies of a binary tree or complete graph. At the same time, between one and three pieces of information were sent out. The Gossip algorithm was working in the SI model. Efficiency of push, pull and hybrid Gossip algorithms has been compared. For the SIR model, hybrid mode and five nodes connected in the complete graph topology, the research was made to measure number of sent Gossip algorithm's packages, depending on the chosen method of selecting the node which is to set information in the Removed state.

To present the operation and scalability of the Gossip algorithm, mathematical model of the algo-

Address for correspondence: Institute of Computer Science, Siedlce University of Natural Sciences and Humanities, 3 Maja54, 08-110 Siedlce, Poland

rithm and graph presenting percentage of infected nodes in individual iterations, is shown. In order to describe the Gossip algorithm using a mathematical formula, a model known from epidemiology was used.

At the end the analysis of research's results was done. As well the problems related to the Gossip algorithm and the method for solving them were described. The application of the Gossip algorithm in commercial solutions was also presented.

1. The problem of information distribution in a distributed systems

1.1. Distributed systems

Nowadays, more and more IT systems are distributed systems or systems that are using cloud solutions. We are calling distributed system the set of computers containing the cohesive software and connected with each other by the network. It contains of few up to few thousands of connected computers called nodes.

Applying distributed systems allows to increase the speed of executing complex algorithms. It is taking place through the division of calculations into a lot of computational processes. These processes, in a concurrent way perform, calculations and exchange information between themselves or only synchronize the results.

Such a kind of systems allow to significantly increase computing power and capitalize the resources of each of the nodes in more efficient way. One should not forget that applying distributed architecture increases systems reliability.

Distributed systems are characterized by several features. One of most important of them is transparency. This feature guarantee that the user of a system is not aware of such a system's parameters like geographical location of each of the nodes or system's size. The breakdown of nodes should as well be not perfectible by the user. Distributed system should guarantee as well the transparency of the methods of access to it. In that case end user has the impression that system is consisting of a single node not many thousands of them. Other feature is the transparency of the transfer, that allows to change position of some resources without the knowledge of the end user. We can deal also with transparency in terms of redoubling. In this case the user is not aware of redoubling certain resources.

Distributed system should be open for cooperation with other systems. It is pretty important that functionality of the distributed systems must be described using very precise interfaces. Another crucial feature of distributed systems is its' scalability taking into consideration both amount of nodes and their geographical location. Other point is the concurrency of the systems that allows to perform many tasks at one time. Easy way to configure and reconfigure is as well important feature of distribution system. One of the most important feature of distributed systems is their ability to share the same resources by all of the users. Their diversity in terms of the hardware is a next essential feature. Ensuring all features mentioned above is not an easy or trivial topic.

Main feature that guarantees the transparency is the ability to share and distribute information inside of a system. One of a most common solutions allowing to distribute information in a smooth way is Gossip algorithm (know as well as Gossip protocol).

1.2. One-to-One Algorithm

While working with huge distributed systems containing of hundred thousands of nodes, we need to deal with a problem of distributing information in a efficient and quick way. It is crucial that data, that was received by one of the nodes, will be send to others in a fast way and without too much usage of network resources. There are several algorithms that help us to deal with that problem. The simplest, however also less efficient, algorithm of spreading information in a distributed systems is algorithm one-to-one. Mentioned algorithm is just sending information node by node. In a first iteration node receiving new piece of information sends it to his neighbor. In the flowing iterations information is distributed node by node. In case of the one-to-one algorithm the path of information flow is created. It leads to the decreasing of algorithms reliability. In case of a crash of network connection between two of the following nodes information is not farther sent. Due to the fact that one node sends information just to the one neighbor full procedure is very time consuming and not efficient. Function of information's propagation time according to number of nodes is lineal and can be described as $t_p = t_i * n$ where t_p is time of propagation, t_i is time of one iteration and n is number of nodes. Assuming that the iteration time is 1 second and number of nodes is 1000 propagation time is 1000 seconds what is about 16 minutes. I would like to admit that the one-to-one algorithm is not scalable, that means that the results will be even worse when the number of nodes increase.

1.3. All-to-All Algorithm

Other algorithm of spreading the information in distributed systems is algorithm all-to-all. In this algorithm each of the node, sends the newly received information to all system's nodes, as soon as he get it. Such a solution allows to increase the reliability of the distributing of information. Compare to one-to-one algorithm crash of connection between two of the nodes does not have such a negative effects. In case of problems with the connection between two of the nodes, information can be sent by different path from other node. The problem with propagation time of the information known from one-to-one algorithm has been solved as well. Full system receive the information almost immediately after it appearance. Unfortunately all-to-all algorithm is not free from flaws and limits. The major flaw is the amount of information sent in the system in each iteration which is N^2 where N is number of nodes. The limit of the nodes in the system derives directly from this flaw. In case of the systems containing tens of thousands of nodes, in each iteration even thousands of millions of information can be send. For both computer network and a single node it is impossible to handle such a network traffic.

1.4. Gossip algorithm

Xerox company at the end of 1980's struggled with a problem of replicating data on few thousands instances of distributed data base. Crucial aspects for Xerox were both time of spreading big amount of information and the reliability of full replication process. One of the propositions was Gossip algorithm called as well Epidemic algorithm. The rule of operation of this algorithm is very similar to spreading either gossip or the disease in some population [1]. Nodes of the system having information sends it to chosen neighbor in a iterative way. The way of selecting the neighbor might be random but it does not need to be such. Each of the nodes might calculate dynamic parameters of connection (like transmission time or quality parameters of transport medium). Based on that parameters for each of nodes there will be assigned a factor. Nodes with higher factor will be prioritized while selecting neighbor for information

sending. For nodes in a system, based on possessing of information, one of three following statuses (states) can be assigned [4]:

1. Infected by the information (I)
2. Susceptible (S). This status is assigned to the nodes not having information.
3. Removed (R). Nodes can be in state R when the information he has is old and is not distributed any more.

There exists two main models of changing of statuses - Model SI and Model SIR. In Model SI there is possibility only for changing state from S to state I, while in SIR Model change from I to R is also allowed. The Gossip algorithm can work in three different modes: push, pull and hybrid push-pull. In case of SI push implementation, nodes being in S state only are passively listening, waiting for new information. In the pull version, nodes with status S assigned as well sends requests for a new piece of information. The hybrid model is combination of Push and Pull. Pseudo code of the Gossip algorithm in SI model is presented in listing 1.

```
1   public void Gossip()
2   {
3     while (true)
4     {
5       Thread.Sleep(timePeriod);
6       Node selectedNode = GetRandomNode();
7       if (this.push)
8       {
9         if (this.state == NodeState.Infected)
10        {
11          SendUpdate(selectedNode, info);
12        }
13
14      }
15      if (this.pull)
16      {
17        SendUpdateRequest(selectedNode);
18      }
19    }
20  }
21
22  private void OnGetUpdate(UpdateArgs args)
23  {
24    args.reciver.informations.Add(args.info);
25    args.reciver.state = NodeState.Infected;
26  }
27
28  private void OnGetUpdateRequest(UpdateArgs args)
29  {
30    if (this.state == NodeState.Infected)
31    {
32      SendUpdate(args.sender,args.info);
```

```
33        }
34      }
```

Listing 1. Gossip Algorithm in SI model

It is wealth mentioning that in SI model for both push and hybrid implementation data spreading will not be stopped even when all the nodes have received information. In case of pull implementation this problem do not occur, as long as all the nodes has information about amount of data that need to be redistributed. If all the nodes receives required number of information, they will stop sanding update requests. Supposing that nodes do not have knowledge about amount of information in a system, pull requests will be as well sent even when system is in consistent state. Constant sending requests and information may lead to overloading of a network. One of the solutions for described problem is SIR model. In SIR model for each piece of information the aging factor is added. After reaching this value information is set into state R and is not sent any more. Specifying aging factor is not an easy task. While doing that we need to take into consideration many aspects, like for example number of nodes and delays in computer network. Underestimation of the factor may lead to not sending information to some of the nodes. Overestimation however cause too big number of information and overloading the network. There are several methods of estimating the aging factor and most of them requires sending the return message after receiving already known information. In first implementation information can be set into status R after being sent redundantly defined number of times. Next one allows to stop sending information, with probability P, after sending every duplicated message. In last implementation one of the nodes having obsolete data is chosen to set the information in status R [3]. Pseudo code of Gossip protocol in a SIR model is presented in Listing 2. The differences in implementation between SI and SIR model were made in lines from 21 to 50.

```
1  public void Gossip()
2  {
3    while (true)
4    {
5      Thread.Sleep(timePeriod);
6      Node selectedNode = GetRandomNode();
7      if (this.push)
8      {
9        if (this.state == NodeState.Infected)
10       {
11         SendUpdate(selectedNode, info);
12       }
13
14     }
15     if (this.pull)
16     {
17       SendUpdateRequest(selectedNode);
18     }
19   }
20 }
21 private void OnGetUpdate(UpdateArgs args)
22 {
23   args.reciver.informations.Add(args.info);
```

```
  args.reciver.state = NodeState.Infected;
  SendResponse(args.sender, informationStatus);
}
private void OnGetResponse()
{
  if (RemovalImplementation.Number == remImplementation && resendedMessages ==
      obsolescence)
  {
    this.state = NodeState.Removed;
  }
  else if (remImplementation == RemovalImplementation.Probability)
  {
    if (GetProbablity())
    {
      this.state =NodeState.Removed;
    }
    else
    {
      resendedMessages++;
    }
  }
}
private bool GetProbablity()
{
  var rand = new Random();
  var p = rand.Next(1, obsolescence);
  return p < obsolescence;
}
private void OnGetUpdateRequest(UpdateArgs args)
{
  if (this.state == NodeState.Infected)
  {
    SendUpdate(args.sender, args.info);
  }
}
```

Listing 2. Gossip Algorithm in SIR model

Other crucial problem, existing in Gossip algorithm, is gathering knowledge about system topology by each of a nodes. In case of huge systems it requires establishing very big list of connections on every node, what might have negative influence on scalability of the Gossip algorithm. Finding out newly joined nodes is as well problematic issue. It requires from nodes sharing information about the topology. Such a procedure generates additional big network traffic. Another solution states that one central managing node, that will have full knowledge about system structure, should be created. It is as well not perfect resolution cause the reliability of a system is much lower. In most efficient implementation every node has just partial information about a system. Assuming that combination of information from all the nodes allows to send information between every nodes it is best solution. More than that information about system topology can be as well sent with Gossip messages.

It is wealth mentioning that Gossip algorithm allows to spread more than one information in a system in the same time. Taking it into consideration it is possible that not all nodes will have the same information, opposite to all-to-all algorithm, full knowledge and coherence will be achieved after few iterations. Gossip algorithm is an algorithm of final coherence, that not guarantee strong coherence of a system. That is why it cannot be used in systems requiring very strict coherence like transactional bank systems or internet stores. Some of the implementation of Gossip algorithm allows to send information to more the one neighbor in the same iteration. It for sure reduce number of iterations and speeds up the process of information sharing, however we need to remember that choosing to big number of neighbors to sent information will lead to creating almost every-to-every algorithm and overloading network.

1.5. The description of a application and lab environment

Algorithm Gossip has been implement using .NET Remoting technology in C# programming language. Application allows to spread more than one information at one time and for changing the Gossip model (SI and SIR) . As well it is possible to specify number of neighbors for which information will be sent in one iteration and the aging factor. In case of SIR model, user is able to select method of choosing nodes that should be assigned to R status. Algorithm Mode can be parametrized as well. Application shows number of sent network packages corresponding with Gossip protocol. As well the state of each node (number of information it has) can be shown. Base on the shown number of information in each iterations the efficiency of a Gossip algorithm is presented. The lab environment contains of ten, connected with network, physical servers (nodes), on which the test application is running. The nodes were connected in two different configurations. In first one the notes are connected into a complete graph while in second one into binary tree.

1.6. Analysis of Gossip Algorithm

The way Gossip algorithm in SI model is working, has been examined for six, eight and ten nodes. For six nodes binary tree topology presented on Fig. 1 was used.

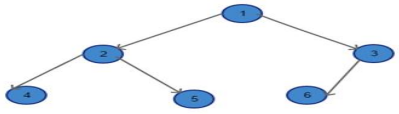


Figure 1. Network topology for six nodes

One or two pieces of information were sent. They were propagated from nodes number one, six or both. When the information was sent from first node, the most efficient occurred to be algorithm Gossip in a hybrid mode. The system was coherent in third iteration. Using push and pull modes, nodes needed five iteration to spread the information. While studying the distribution of messages from node number six, we will also see the highest effectiveness using the hybrid mode. All nodes received information in the fifth iteration. In the push mode information was distributed in sixth iteration, while in pull mode not until eighth. We can observe similar results, when the information's packages were sent from both node one and six at ones. Using hybrid mode algorithm was able to spread information in three iterations. For push and pull six iterations were needed to achieve such a result. For six nodes, as

well examination of spreading information in system of complete graph topology was done. One or two pieces of information were sent. In both cases most efficient occurred to be hybrid mode. For both one and two pieces of information system get coherent in second iteration. In pull mode it took place in third iteration for one information and forth for two pieces of information. In both cases push mode needed four iteration to spread information over the system. Analogous tests were made for a system containing eight nodes. Binary tree topology of connections between servers is presented on Fig. 2.

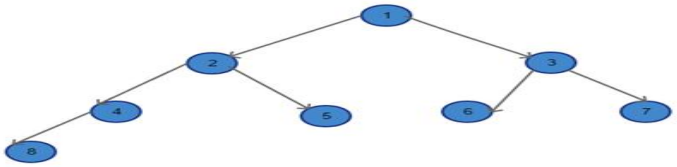


Figure 2. Network topology for eight nodes

For hybrid and pull modes, when infected node was that with number one, information was distributed in second iteration. Algorithm Gossip in push mode needed six iteration to achieve it. For information distributed from node number eight, in hybrid mode, system get coherent in sixth iteration. Using pull method, algorithm was able to spread information in seven iterations, while using push in twelfth. Distributing two information's packages (one from node seven and one from node eight) lead to worse results. Gossip protocol in a push mode needed up to twenty seven iteration to spread information. Pull method was able to redistribute information in eleven iteration and hybrid in nine. For hybrid mode sending three pieces of information was tested. They were distributed from nodes number eight, seven and six. Coherence was achieved in eighth iteration. Next measurements for nodes connected into complete graph were done. Spreading one information took four iterations for push, six for pull and three for hybrid. When number of data packages was extended to two, the most efficient occurred to be hybrid and push modes. In both cases system was coherent in forth iteration. Pull mode needed one iteration more to achieve that. For hybrid mode the number of selected neighbors, for information distribution in one iteration, was extended to two. It effected in spreading one information in two iterations. Similar experiments were made for system containing ten nodes. The topology of network connections between servers is presented on Fig. 3.

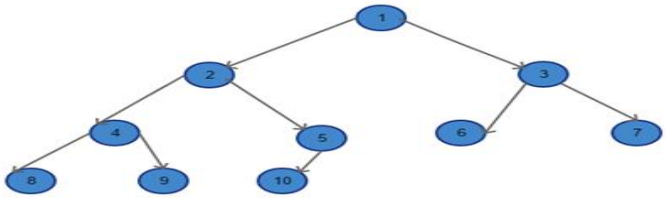


Figure 3. Network topology for ten nodes

When information was sent from node number one in push mode, the system achieved coherency in the fourteenth iteration. In the pull mode, this took place in the ninth iteration, while in the hybrid mode in the fifth. Considering the first node infected was node number ten, the Gossip algorithm, in

both push and pull mode, spread the data in eleven iterations. In hybrid mode, each server received information after six iterations. For two pieces of information sent from nodes nine and ten in push mode, the broadcast occurred in seven passes. For the pull mode this was done in the fifteenth iteration and for the hybrid mode in the fifth. After connecting all ten nodes in a complete graph topology, the results are much better. For one piece of information sent in the push mode, the system was coherent after the fourth iteration. In the pull mode it took three more iterations, while in the hybrid mode the message was sent within two iterations. In the case of sending two messages in the push mode, the information was disseminated during six iterations, in the pull mode during seven, while in the hybrid mode during three. In the SIR model, for five nodes connected in topology of complete graph, hybrid mode and one message, tests of various model configurations were made. In the SI model, every node within forty-five iterations sends ninety-two packages related to the exchange of information (requests to send a message and the messages themselves). With a larger number of nodes, it can be a significant load for the network. In order to minimize the number of requests and information sent, an SIR model was introduced, allowing to stop broadcasting information under certain conditions. The behavior of the Gossip algorithm in three variants of the SIR mode was investigated. The first was to stop sending messages with a set probability P / the probability of changing the status of information from the state I to R / at the moment when the neighboring node sends information that it already received particular message before. The tests began with a P value of 0.05. For this configuration, nodes sent from seventy-seven to one hundred and five packages during forty-five iterations. Considering that this value also contains the amount of feedback messages and that number of network packages is smaller than in the SI model, the effect is satisfying. Then the value of P was increased to 0.1. The result of the increase was visible and resulted in a reduction in the number of packages from individual nodes to a number between sixty and seventy. After another increase in the value of P to 0.2, the number of packages sent from one node was reduced to a value between fifty and fifty-seven. A further increase in the value of P did not bring such large changes. For P equal to 0.3, nodes send out fifty to fifty-five packages, while for a P of 0.5, the result is between forty five and fifty packages. For the higher probability values, it was already possible to observe the lack of final system coherency, cause the information was set in R status on all nodes before it was sent out over the entire system. Another implementation of the Gossip algorithm in the SIR model assumes switching to the R state after sending n redundant messages. Assuming n equal three, each node sent fifty to fifty three packages. The results for n equal to five are identical. At n of six, this number increased to fifty four - fifty-seven packages. Assuming n equal to eight, we can observe another increase in the number of packages to a value between fifty-five and sixty. The next implementation allowed sending by a predetermined management node to a randomly selected neighbor containg a message, request to change the information's state to R. In this implementation, each node sent within seventy messages. A summary of the test results for the number of packages sent is presented in Tab. 1.

The researches were made on a relatively small number of nodes, so the question arises about the efficiency of the algorithm for much larger systems. In the case of a complete graph topology, it can be assumed that the probability p of choosing each of the n neighbors is equal to $1/n$. The Gossip algorithm of the SI model and in the push mode can be described using a mathematical model known from epidemiology, showing the number of infected people (in our case nodes) depending on the time. The formula for the number of infected nodes is shown below [6]:

$$U(t) = \frac{n+1}{1 + n * e^{-(n+1)*p*t}}$$

Table 1. Number of sent packages in Gossip algorithm

	Minimal number of packages	**Maximal number of packages**
SI		
	92	92
SIR		
With Probability P		
0.05	77	105
0.1	60	70
0.2	50	57
0.3	50	55
0.5	45	50
After n redundant messages		
3	50	53
5	50	53
6	54	57
8	55	60
Random	68	73

where: n is the number of nodes, and t is the number of iterations. Assuming that the number of nodes is one hundred thousand, we need only twenty-four iterations to achieve coherency. For a million nodes, the number of iterations only increases to twenty-eight. For the pull and hybrid models the results should be even better. Fig. 4 shows the percentage share of infected nodes in iterations for ten, one hundred, one thousand, ten thousand one hundred thousand and million nodes.

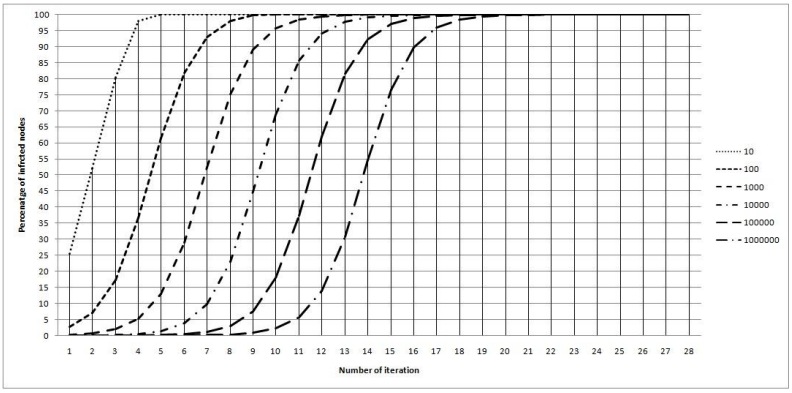


Figure 4. Percentage of infected nodes In each of the iterations

The chart shows that after the seventeenth iteration system coherence for the number of nodes less than a million is more than ninety percent, and from the nineteenth iteration more than ninety-nine percent.

2. Summary

Based on the results of the tests carried out and the calculations made, it can not be argued that the Gossip algorithm is perfectly suited for the spreading of information in distributed systems. However, it should be taken into account that the effectiveness of the algorithm depends on many factors. The topology of connections between nodes has a fundamental influence on the operation of the gossip algorithm. In hierarchical structures, which is examined tree topology, information needs more iteration to reach all the nodes than in topology of complete graph. In the tree structure, the node from which the information will be sent is not without significance. The deeper the infected server is in the network structure, the more iterations are needed to bring about system integrity. The choice of algorithm's operating modes also affect on the efficiency of its work. The fastest mode is hybrid mode. Slightly worse results can be observed for the pull mode. The weakest effects gives the use of the push mode. The hybrid mode, unfortunately, is not free of drawbacks. The main one consists in sending a large number of network packages. Each of the nodes operating in this mode, during each iteration, sends out one to two packages. The first package is responsible for sending a request message. The second is sent if the table of the information is not empty.The amount of packages sent can be significantly reduced using the SIR algorithm of the Gossip algorithm. It allows not to broadcast information that has already been sent to many nodes. There are several implementations that allow to set R status for the information, causing the information not to be distributed. The most effective implementation was the transition to the status of R with a constant probability P, in the case of obtaining message about redundant information. For

the probability P from 0.2-0.5, the number of packages sent has been halved. Implementation allowing to determine the status R after sending redundant message n times gives slightly worse results. Random selection of nodes to set the R status for information is the least effective and seems to be the most risky of all described approaches. In specific situations incorrect selection of the node may cause the system will not achieve the final coherence. This happens when the node being drawn is the only one that has a connection to a part of the system and will not be able to send information there. I think that the interesting aspect of the Gossip algorithm is its scalability. In the hybrid mode and the complete graph topology, I did not notice a significant difference in the number of needed iterations while sending one message for six nodes and sending two messages to ten nodes. During the study of the Gossip protocol in the push mode, in particular in the hierarchical network topology, one could notice a large dependence of the protocol's effectiveness on the selection of neighboring nodes when sending a message. In the case of two neighbors, the probability of drawing each of them is equal to 0.5. It may happen that in the course of several iterations, the same neighbor will be drawn. This will result in the information not reaching the other of the adjacent nodes and extending the information transfer process. The solution to this problem is to create a local dictionary to determine for which nodes the information was sent. This would eliminate, from the drawing, the nodes that received the message. The described problem does not occur when we have many adjacent nodes, because the probability of drawing each of them becomes smaller as the number of connected nodes increases. It is obvious that during the implementation of the Gossip algorithm special attention should be paid to the problems of mutual distributed exclusion. Exclusion can occur when multiple nodes at the same time try to send a message to the same node. The solution may be to use an algorithm that solves this problem, such as the Ricard Agrawal's algorithm based on requests. It should be mentioned that the Gossip algorithm is highly scalable, as shown by mathematical methods. This favors the popularity of solutions based on the this algorithm. It is used, among others, in the Apache Cassandra software, which is a free tool for managing no-SQL distributed databases. Another system using the epidemiological algorithm is SERF [2], used to manage server farms. The Gossip protocol has also been applied to Amazon Web Services [5].

References

[1] A. Dcmers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenkcr, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance.

[2] HashiCorp. Gossip protocol.

[3] M. Jelasity. Gossip-based protocols for large-scale distributed systems.

[4] F. Lopez. Introduction to gossip.

[5] The Amazon S3 Team. Amazon s3 availability event: July 20, 2008.

[6] S. Zborowski. Algorytmy w chmurach. część 1: Gossip. *Programista*, 44(1), 2016.

Extensions of Elementary Cause-Effect Structures
Extended abstract

Ludwik Czaja[1,2]

[1] Vistula University
[2] University of Warsaw

Before formulation of some extensions of elementary cause-effect (c-e) structures (see References), let us outline their concept by examples. A c-e structure (both elementary - a counterpart of 1-safe Petri nets - and extended) is a directed graph with predecessors and successors of every vertex (node) grouped into families of sets, as shows left graph in Fig.1: predecessors of e: $\{\{a,b\},\{b,c\},\{d\}\}$, successors: $\{\{f,g\},\{h\}\}$. In the right graph, the node sym-

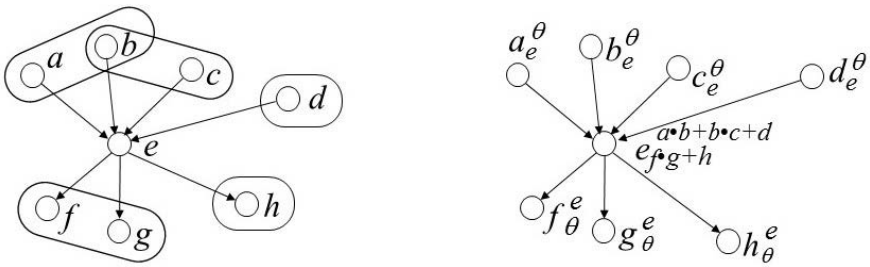


Fig. 1. left: graph with encircled families of predecessors and successors of e; right: graph with subscripted and superscripted symbols of nodes.

bols are subscripted and superscripted with expressions called formal polynomials, that determine the grouping, so that the "operator of multiplication $\bullet$" collects the arguments into a group, whereas "operator of addition +" separates the groups. Symbol θ denotes the empty family. Thus, this graph is the set $\{a_e^\theta, b_e^\theta, c_e^\theta, d_e^\theta, e_{f\bullet g+h}^{a\bullet b+b\bullet c+d}, f_\theta^e, g_\theta^e, h_\theta^e\}$. Each c-e structure can be represented by a set of such annotated nodes. The arrows, though helpful to understand its dynamics, are a superfluous information. Informally, the dynamics is a "token game": node e can receive signals (represented by tokens) simultaneously from a and b or simultaneously from b and c or only from d, and send signals simultaneously to f and g or only to h. Thus, the operator "$\bullet$" means simultaneity, while "+" - exclusive choice. As a realistic example, consider the c-e structure *ROAD* in Fig.2, describing a traffic through the bridge B on the two-lane road, each lane for vehicles heading in the opposite directions. The bridge can hold only one vehicle at a time.

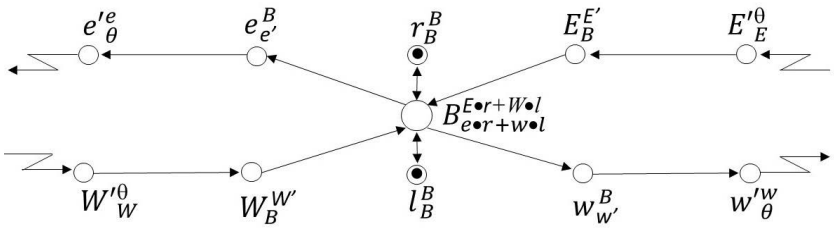


Fig. 2. c-e structure *ROAD*. Traffic from the East: $E' \to E \to B \to e \to e'$ and from the West: $W' \to W \to B \to w \to w'$. Nodes r and l prevent the U-turn on the bridge: a token in r makes impossible move $E \to B \to w$, while in l - impossible move $W \to B \to e$.

Thus, in the set-like notation,
$ROAD = \{E'^{\theta}_{E},\ E^{E'}_{B},\ B^{E\bullet r+W\bullet l}_{e\bullet r+w\bullet l},\ r^{B}_{B},\ e^{B}_{e'},\ e'^{e}_{\theta},\ W'^{\theta}_{W},\ W^{W'}_{B},\ l^{B}_{B},\ w^{B}_{w'},\ w'^{w}_{\theta}\}$.
Anticipating the formal definitions, notice that this is a combination $ROAD = EW \bullet R + WE \bullet L$, where $EW = \{E'^{\theta}_{E}, E^{E'}_{B}, B^{E}_{e}, e^{B}_{e'}, e'^{e}_{\theta}\}$,
$WE = \{W'^{\theta}_{W}, W^{W'}_{B}, B^{W}_{w}, w^{B}_{w'}, w'^{w}_{\theta}\}$, $r = \{r^{B}_{B}, B^{r}_{r}\}$, $l = \{l^{B}_{B}, B^{l}_{l}\}$, or pictorially, a combination of the c-e structures in Fig.3. So, the "multiplication" and "ad-

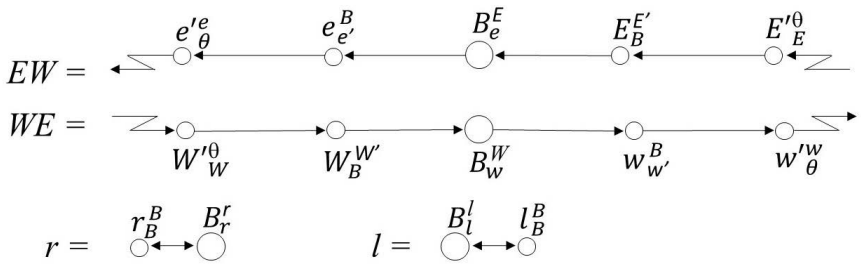


Fig. 3. traffic East → West and West → East, and no-U-turn control.

dition" are now extended from the formal polynomials onto c-e structures, so that "•" and "+" mean making union of sets being their arguments, with formal product and sum of subscripts/superscripts of nodes identically named in both sets. Now, the formal definitions.

Definition 1 (set $\boldsymbol{F}[\mathbb{X}]$, quasi semiring of formal polynomials)

Let $\mathbb{X}$ be a non-empty enumerable set. Their elements, called *nodes*, are counterparts of places in Petri nets [Rei 85]. Let $\theta \notin \mathbb{X}$ be a symbol called *neutral*. It will play a role of neutral element for operations on expressions, called *formal polynomials over* $\mathbb{X}$. The names of nodes, symbol θ, operators $+$, $\bullet$ and parentheses are symbols out of which formal polynomials are formed in the usual

(infix) way. Their set is denoted by $\boldsymbol{F}[\mathbb{X}]$. Stronger binding of $\bullet$ than $+$, allows for dropping some parentheses. Addition and multiplication of $K, L \in \boldsymbol{F}[\mathbb{X}]$ is defined as follows: $K \oplus L = (K + L)$, $K \odot L = (K \bullet L)$. Let us use $+$ and $\bullet$ instead of $\oplus$ and $\odot$. It is required that the system $\langle \boldsymbol{F}[\mathbb{X}], +, \bullet, \theta \rangle$ obeys the following equality axioms for all $K, L, M \in \boldsymbol{F}[\mathbb{X}]$, $x \in \mathbb{X}$:

$(+)$	$\theta + K = K + \theta = K$	$(\bullet)$	$\theta \bullet K = K \bullet \theta = K$
$(++)$	$K + K = K$	$(\bullet\bullet)$	$x \bullet x = x$
$(+++)$	$K + L = L + K$	$(\bullet\bullet\bullet)$	$K \bullet L = L \bullet K$
$(++++)$	$K + (L + M) = (K + L) + M$	$(\bullet\bullet\bullet\bullet)$	$K \bullet (L \bullet M) = (K \bullet L) \bullet M$
$(+\bullet)$	If $L \neq \theta \Leftrightarrow M \neq \theta$ then $K \bullet (L + M) = K \bullet L + K \bullet M$		

Algebraic system which obeys these axioms will be referred to as a *quasi-semiring of formal polynomials.*[3] □

The system $\langle \boldsymbol{F}[\mathbb{X}], +, \bullet, \theta \rangle$ has a "family of sets" model shown above, thus is consistent. Its peculiarity, in contrast to the ordinary semiring, is axiom $(+\bullet)$ - the conditional distributivity of multiplication over addition, and the neutral θ for both operations. These assumptions make c-e structures behaviourally equivalent to Petri nets.

Definition 2 (cause-effect structure, carrier, set $\boldsymbol{CE}$)

A cause-effect structure (c-e structure) over $\mathbb{X}$ is a pair $U = (C, E)$ of total and injective functions:

C: $\mathbb{X} \to \boldsymbol{F}[\mathbb{X}]$ (*cause function*; nodes occuring in $C(x)$ are *causes* of x)
E: $\mathbb{X} \to \boldsymbol{F}[\mathbb{X}]$ (*effect function*; nodes occuring in $E(x)$ are *effects* of x)

such that x occurs in the formal polynomial $C(y)$ iff y occurs in $E(x)$. *Carrier* of U is the set $car(U) = \{x \in \mathbb{X} : C(x) \neq \theta \vee E(x) \neq \theta\}$. U is of *finite carrier* iff $|car(U)| < \infty$ ($|\ ...\ |$ denotes cardinality). The set of all c-e structures over $\mathbb{X}$ is denoted by $\boldsymbol{CE}[\mathbb{X}]$. Since $\mathbb{X}$ is fixed, we write $\boldsymbol{CE}$ - wherever this makes no confusion. □

Since functions C and E are total, each c-e structure comprises all nodes from $\mathbb{X}$, also the isolated ones - those from outside of its carrier. Presenting c-e structures graphically, only their carriers are pictured.

Definition 3 (addition and multiplication, monomial c-e structure)

For c-e structures $U = (C_U, E_U)$, $V = (C_V, E_V)$ define:
$U + V = (C_{U+V}, E_{U+V}) = (C_U + C_V, E_U + E_V)$ where
$(C_U + C_V)(x) = C_U(x) + C_V(x)$
$U \bullet V = (C_{U \bullet V}, E_{U \bullet V}) = (C_U \bullet C_V, E_U \bullet E_V)$ where
$(C_U \bullet C_V)(x) = C_U(x) \bullet C_V(x)$

[3] In the early papers on cause-effect structures, the term "near-semiring" has been used. But in the meantime some authors used it in different meaning, so, we use term "quasi-semiring" for this axiomatic system.

(The same symbol is used for multiplication of c-e structures, functions and polynomials)

U is a *monomial* c-e structure iff each polynomial $C_U(x)$ and $E_U(x)$ is a monomial, i.e. does not comprise non-reducible "+". □

Evidently $U + V \in \boldsymbol{CE}$ and $U \bullet V \in \boldsymbol{CE}$ that is, in the resulting structures, x occurs in $C_{U+V}(y)$ iff y occurs in $E_{U+V}(x)$ and the same for $U \bullet V$. Thus, addition and multiplication of c-e structures yield correct c-e structures.

The set $\boldsymbol{CE}$ with addition, multiplication and a distinguished element denoted also by θ and understood as the empty c-e structure (θ, θ), where θ is a constant function $\theta(x) = \theta$ for all $x \in \mathbb{X}$, makes an algebraic system similar to that in Definition 1.

Proposition 1 (quasi semiring of c-e structures)

The system $\langle \boldsymbol{CE}[\mathbb{X}], +, \bullet, \theta \rangle$ obeys the following equations for all $U, V, W \in \boldsymbol{CE}[\mathbb{X}]$, $x, y \in \mathbb{X}$:

(+)	$\theta + U = U + \theta = U$	(•)	$\theta \bullet U = U \bullet \theta = U$
(++)	$U + U = U$	(••)	$(x \to y) \bullet (x \to y) = x \to y$
(+++)	$U + V = U + V$	(• • •)	$U \bullet V = V \bullet U$
(++++)	$U + (V + W) = (U + V) + W$	(••••)	$U \bullet (V \bullet W) = (U \bullet V) \bullet W$
(+•)	If $C_V(x) \neq \theta \Leftrightarrow C_W(x) \neq \theta$ and $E_V(x) \neq \theta \Leftrightarrow E_W(x) \neq \theta$ then $U \bullet (V + W) = U \bullet V + U \bullet W$		

□

This follows directly from definition of c-e structures and definitions of adding and multiplying c-e structures. The operations on c-e structures make possible to combine small c-e structures into large ones.

Definition 4 (partial order $\leq$; substructure, set $\boldsymbol{SUB}[V]$)

For $U, V \in \boldsymbol{CE}$ let $U \leq V \Leftrightarrow V = U + V$; obviously, $\leq$ is a partial order in $\boldsymbol{CE}$. If $U \leq V$ then U is a *substructure* of V; $\boldsymbol{SUB}[V] = \{U : U \leq V\}$ is the set of all substructures of V. For $A \subseteq \boldsymbol{CE}$: $V \in A$ is *minimal* (w.r.t. $\leq$) in A iff $\forall W \in A$: $(W \leq V \Rightarrow W = V)$ □

The crucial notion for behaviour of c-e structures is firing component, a counterpart of transition in Petri nets, i.e. a state transformer. It is, however, not a primitive notion but derived from the definition of c-e structures, and is introduced regardless of any particular c-e structure:

Definition 5 (firing component, set $\boldsymbol{FC}$, pre-set and post-set)

A minimal in $\boldsymbol{CE} \backslash \{\theta\}$ c-e structure $Q = (C_Q, E_Q)$ is a *firing component* iff Q is a monomial c-e structure and $C_Q(x) = \theta \Leftrightarrow E_Q(x) \neq \theta$ for any $x \in car(Q)$. The set of all firing components is denoted by $\boldsymbol{FC}$, thus the set of all firing

components of $U \in \boldsymbol{CE}$ is $\boldsymbol{FC}[U] = \boldsymbol{SUB}[U] \cap \boldsymbol{FC}$. Following the standard Petri net notation, let for $Q \in \boldsymbol{FC}$ and $G \subseteq \boldsymbol{FC}$:

${}^\bullet Q = \{x \in car(Q) : C_Q(x) = \theta\}$ (*pre-set* of Q)
$Q^\bullet = \{x \in car(Q) : E_Q(x) = \theta\}$ (*post-set* of Q)
${}^\bullet Q^\bullet = {}^\bullet Q \cup Q^\bullet$ □

So, the firing component is a connected graph, due to the required minimality. Elements of the pre-set are its *causes* and elements of the post-set are its *effects*. Of many conclusions from above definitions, some are worth to point out:

Proposition 2

(a) $U_1 \leq V_1 \wedge U_2 \leq V_2 \Rightarrow U_1 + U_2 \leq V_1 + V_2$ (monotonicity of $+$)
(b) $U \bullet (V + W) \leq U \bullet V + U \bullet W$ but equality not always holds
(c) $U \leq V \Rightarrow \boldsymbol{FC}[U] \subseteq \boldsymbol{FC}[V]$ but converse implication not always holds
(d) $\boldsymbol{FC}[U] \cup \boldsymbol{FC}[V] \subseteq \boldsymbol{FC}[U + V]$ but converse inclusion not always holds

Point (d) states that new firing components may appear when summing up c-e structures. For instance, let $U = \{a_{x+y}, b_{x\bullet y}, x^{a\bullet b}, y^{a\bullet b}\}$, $V = \{a_{x\bullet y}, x^a, y^a\}$, thus $\boldsymbol{FC}[U] = \emptyset$, $\boldsymbol{FC}[V] = \{V\}$,
$\boldsymbol{FC}[U + V] = \{\{a_x, x^a\}, \{a_y, y^a\}, V, \{a_{x\bullet y}, b_{x\bullet y}, x^{a\bullet b}, y^{a\bullet b}\}\}$, thus
$\boldsymbol{FC}[U] \cup \boldsymbol{FC}[V] \neq \boldsymbol{FC}[U + V]$. The phenomenon of creation new firing components when assembling c-e structures from smaller parts, reflects a general observation: compound systems may sometimes reveal behaviours absent in their parts.

Definition 6 (state of c-e structure)

A state of c-e structure U is a total injective function $s : car(U) \to \mathbb{N}_\omega$, thus a multiset over $car(U)$ ($\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$, where ω symbolises infinity, that is $\omega > n$ for each $n \in \mathbb{N}$; $\mathbb{N}$ is the set of natural numbers, with 0). The set of all states of U is denoted by $\mathbb{S}$. □

Definition 7 (weights of monomials and capacity of nodes)

Given a c-e structure $U = (C, E)$ and its firing component $Q \in \boldsymbol{FC}[U]$, let along with the pre-set ${}^\bullet Q$ and post-set $Q^\bullet$ of Q, some multisets $\overline{{}^\bullet Q}$: ${}^\bullet Q \to \mathbb{N}_\omega \backslash \{0\}$ and $\overline{Q^\bullet}$: $Q^\bullet \to \mathbb{N}_\omega \backslash \{0\}$ be given as additional information. The value $\overline{{}^\bullet Q}(x)$ is called a *weight* (or *multiplicity*) of monomial $E_Q(x)$ and the value $\overline{Q^\bullet}(x)$ - a *weight* (or *multiplicity*) of monomial $C_Q(x)$. Let *cap* be a total injective function $cap : car(U) \to \mathbb{N}_\omega$, assigning a *capacity* to each node in the set $car(U)$. A c-e structure with such enhanced firing components is called a *c-e structure-with-weights of monomials and capacity of nodes*. □

Definition 8 (firing components enabled and with inhibitors)

For a firing component $Q \in \boldsymbol{FC}[U]$, the set $inh[Q] = \{x \in {}^\bullet Q : \overline{{}^\bullet Q}(x) = \omega\}$ is the collection of nodes in the pre-set of Q, whose effect monomials $E_Q(x)$ are of weight ω. The nodes in $inh[Q]$ will play role of *inhibiting nodes* of firing component Q, as follows. For Q and state s let us define the formula: $enabled[Q](s) \stackrel{\text{def}}{\Leftrightarrow}$

$\forall x \in inh[Q] : s(x) = 0 \wedge$
$\forall x \in {}^\bullet Q \backslash inh[Q] : \overline{{}^\bullet Q}(x) \leq s(x) \leq cap(x) \wedge$
$\forall x \in Q^\bullet : \overline{Q^\bullet}(x) + s(x) \leq cap(x)$ □

So, Q is enabled at the state s iff none of inhibiting nodes $x \in {}^\bullet Q$ contains a token and each remaining node in ${}^\bullet Q$ does, with no fewer tokens than is the weight of its effect monomial $E_Q(x)$ and no more than capacity of each $x \in {}^\bullet Q$. Moreover, none of $x \in Q^\bullet$ holds more tokens than their number, when increased by the weight of its cause monomial $C_Q(x)$, exceedes capacity of x. The inhibiting nodes of a firing components will be called its *inhibitors*.

Fig.4(a) shows a firing component Q with weighted (multiplied) effect monomials $E_Q(a) = 5 \otimes x$, $E_Q(b) = \omega \otimes (x \bullet y)$, $E_Q(c) = 3 \otimes y$ and weighted cause monomials $C_Q(x) = 2 \otimes (a \bullet b)$, $C_Q(y) = 4 \otimes (b \bullet c)$. The inhibitor of Q is node b. Fig.4(b) shows the behaviourally equivalent single transition in Petri net with weights and inhibitor arrow.

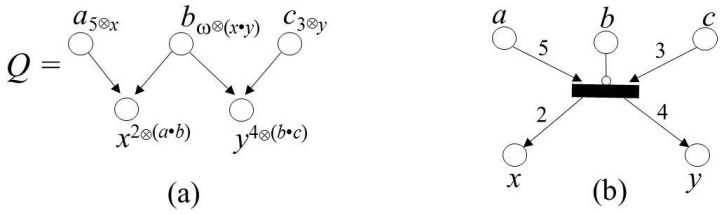


Fig. 4. (a) Firing component Q with weights; $2 \otimes (a \bullet b)$, $\omega \otimes (x \bullet y)$, etc. denote multiplicity of the product $a \bullet b$ by the factor $2 = \overline{Q^\bullet}(x)$ and product $x \bullet y$ by factor $\omega = \overline{{}^\bullet Q}(b)$. (b) Behaviourally equivalent Petri net transition.

Definition 9 (semantics [[ ]] of c-e structures with inhibitors)

For $Q \in \boldsymbol{FC}[U]$, let $[[Q]] \subseteq \mathbb{S} \times \mathbb{S}$ be a binary relation defined as:
$(s,t) \in [[Q]]$ iff $enabled[Q](s) \wedge t = (s - \overline{{}^\bullet Q}) + \overline{Q^\bullet} \leq cap$ (Q transforms state s into t). *Semantics* $[[U]]$ of $U \in \boldsymbol{CE}$ is $[[U]] = \bigcup_{Q \in \boldsymbol{FC}[U]} [[Q]]$. Closure, reachability and computation: $(s,t) \in [[U]]^*$ iff $s = t$ or there is a sequence of states $s_0, s_1, ..., s_n$ with $s = s_0$, $t = s_n$ and $(s_j, s_{j+1}) \in [[U]]$ for $j = 0, 1, ..., n-1$. State t is *reachable* from s in semantics [[ ]] and the sequenece $s_0, s_1, ..., s_n$ is a *computation* in U. □

In the c-e structure which presents a ride throught the bridge B, the priority ride from the East can be enforced using inhibitor, i.e. node E in the pre-set of firing component $\{W_B^\theta, E_{\omega \otimes B}^\theta, l_B^\theta, B_\theta^{W \bullet E \bullet l}\}$, as shown in Fig.5.

Firing components $\{E_B^\theta, r_B^\theta, B_\theta^{E \bullet r}\}$ and $\{W_B^\theta, E_{\omega \otimes B}^\theta, l_B^\theta, B_\theta^{W \bullet E \bullet l}\}$ of the c-e structure in Fig.5 have Petri nets (with inhibitor arcs) counterparts as two transitions shown in Fig.6.

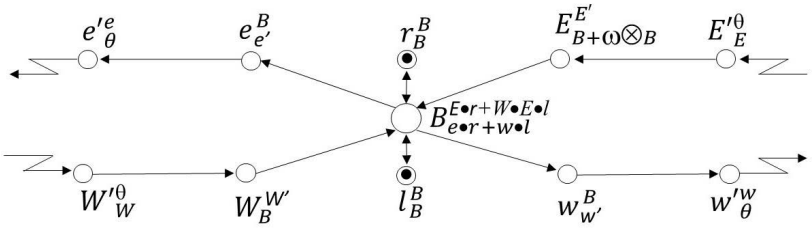


Fig. 5. If at E and W are vehicles (tokens) and none at B, then only the one in E will get entry permit at B, since only firing component $\{E^{\theta}_{B}, r^{\theta}_{B}, B^{E\bullet r}_{\theta}\}$ can fire in such state, not this one: $\{W^{\theta}_{B}, E^{\theta}_{\omega\otimes B}, l^{\theta}_{B}, B^{W\bullet E\bullet l}_{\theta}\}$ - due to its inhibiting node E if it contains a token.

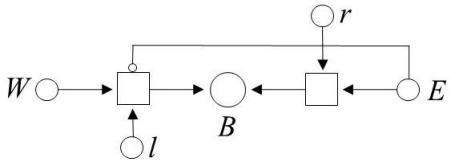


Fig. 6. Petri net counterpart of two firing components with place B of c-e structure shown in Fig.3.2. Inhibitor arc leads from place E to the left transition.

Example (the Readers/Writers problem)

A set of n sequential agents run concurrently under constraint: writing to a common file by the jth ($j = 1, 2, ..., n$) agent prevents all remaining from reading and writing, but not from their private (internal) activity. Reading may proceed in parallel. Fig.7 shows three agents with the following meaning of nodes: Aj - agent of number $j = 1, 2, 3$ is active (holds a token) if it is neither reading nor writing; Rj - is active if the jth agent is reading; Wj - is active if the jth agent is writing. Wj and Rj play both roles: of the ordinary nodes or of the inhibitors, dependently which firing component they belong to.

A few semantic properties of c-e structures are in:

Proposition 3

For any c-e structures $U, V \in \boldsymbol{CE}$:

(a) $U \leq V \Rightarrow \boldsymbol{FC}[U] \subseteq \boldsymbol{FC}[V] \Rightarrow [[U]] \subseteq [[V]] \Rightarrow [[U]]^* \subseteq [[V]]^*$
(b) $[[U]] \cup [[V]] \subseteq [[U + V]]$, but the reverse inclusion not always holds
(c) $\boldsymbol{FC}[U] \cup \boldsymbol{FC}[V] = \boldsymbol{FC}[U + V] \Rightarrow [[U]] \cup [[V]] = [[U + V]]$ but not conversely.
(d) $\boldsymbol{FC}[U] \cup \boldsymbol{FC}[V] = \boldsymbol{FC}[U + V]$ and $[[U]]^* \cup [[V]]^* = [[U + V]]^*$
are unrelated by implication. □

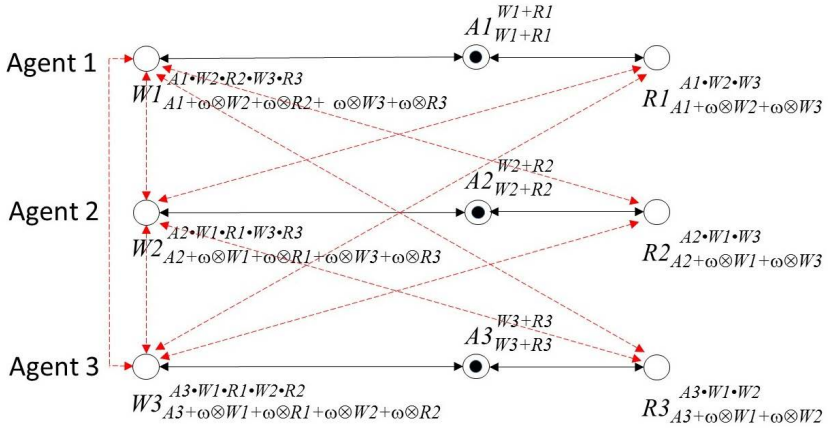


Fig. 7. Three agents' Readers/Writers system as a c-e structure RW with inhibitors; the dashed arrows denote usage of inhibitors. Initially, the agents are neither reading nor writing (tokens in $A1, A2, A3$).

Another extension: c-e structures with time.

Time models are different from those in Petri nets with time, where time is usually treated as necessary or admissible period of activity of a node (site or action). Here, the minimal time model is considered, where capacity of nodes equal 1, and with each node a time period of ***mandatory*** stay of a token is associated. This is the shortest time during which the node ***must*** hold the token. On expiry of this period, the token ***can*** leave the node (if other necessary conditions for this "move" are met). Lapse of time may be related either to individual firing components or to the whole c-e structure. The period of a token stay at a node is set up on entering this token into it and decreases by one time unit ("tick") of the timer referred to by the node, until permission to leave this node. On expiry of the mandatory residing time at this node, the token can leave it if all other conditions for this action are met. Any c-e structure with the minimal time model can be simulated ("implemented") by a c-e structure without time but with some additional nodes associated with every original node. A number of these supplementing and linearly ordered nodes represent duration of mandatory stay of a token in the original node.

Definition 10 (min-time c-e structure, set $T_{min}\,\boldsymbol{CE}$)

$U = \langle C, E, T_{min}\rangle$ is a *minimal-time* c-e structure iff $\langle C, E\rangle$ c-e structure with capacity of nodes equal 1, and T_{min}: $car(U) \to \mathbb{N}\backslash\{0\}$ is a *minimal time function* of the meaning: $T_{min}(x)$ is the least number of time units indicated by a timer referred to by the node x, during which a token ***must*** stay at x since its appearance. The timer is associated to a particular node. The set of all the min-time c-e structures over is denoted by $T_{min}\,\boldsymbol{CE}$ □

Definition 11 (state of min-time c-e structure)

State is a function $s : \mathbb{X} \to \mathbb{N}$ with the informal meaning: $s(x) = 0$ if there is no token at the node x and $s(x) > 1$ is a remaining time (a number of ticks of the timer referred to by x) during which the token ***must*** remain at x; $s(x) = 1$ indicates that the time of ***compulsory*** residence of a token at the node x, prescribed by $T_{min}(x)$ has elapsed, thus, the token ***can*** be moved further - if other conditions for this are satisfied. The set of all states is $\mathbb{S} = \mathbb{N}^{\mathbb{X}}$ (state-space). □

So, now, the $s(x)$ is not a number of tokens residing at the node, but a current time lapse.

Definition 12 (min-time semantics: a firing rule)

For $Q \in \boldsymbol{FC}[U]$, let $[[Q]] \subseteq \mathbb{S} \times \mathbb{S}$ be a binary relation defined as: $(s,t) \in [[Q]]$ if and only if:

$\forall x \in {}^\bullet Q : [s(x) = 1 \wedge t(x) = 0 \wedge \forall y \in Q^\bullet : [s(y) = 0 \wedge t(y) = T_{min}(y)] \vee$
$\exists x \in {}^\bullet Q^\bullet : [s(x) > 1 \wedge t(x) = s(x) - 1]$

Semantics $[[U]]$ of $U \in T_{min}CE$ is the union of relations $[[U]] = \bigcup_{Q \in FC[U]} [[Q]]$.

$[[U]]^*$ is the reflexive and transitive closure of $[[U]]$ □

The formula $\exists x \in {}^\bullet Q^\bullet : [s(x) > 1 \wedge t(x) = s(x) - 1]$ expresses decrease by one time unit of token's stay at a certain node x of Q, if the minimal time of this token has not expired in the state s. The minimal time can be simulated by c-e structures without time constraints. An exemplary simulation of the c-e structure in Fig.8 (firing component) depicts Fig.9.

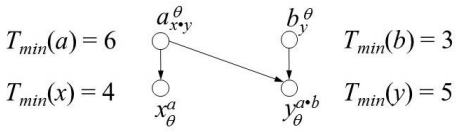


Fig. 8. c-e structure (a firing component) with min-time assigned to nodes.

If the time is taken from a timer common to all nodes (violation of distributed systems' principles!), the semantics is re-interpreted: the decreasing elapse of time now concerns all nodes in $car(U)$, not only a given firing component. Thus, the formula $\exists x \in {}^\bullet Q^\bullet : [s(x) > 1 \wedge t(x) = s(x) - 1]$ would be replaced with $\exists x \in car(Q) : [s(x) > 1 \wedge t(x) = s(x) - 1]$. An example of this case, taken from music, is in Fig.10.

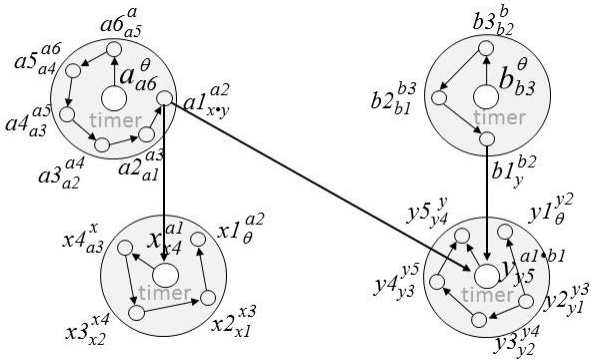


Fig. 9. A possible simulation of the c-e structure in Fig.8 with the minimal time of nodes, by means of no-time c-e structure. The separate timer (each with perhaps diverse progress rate of time) is associated with every node. The counterclockwise direction of a token's motion inside the timers, simulates elapse of time controlled by the timers associated with nodes a, b, x, y.

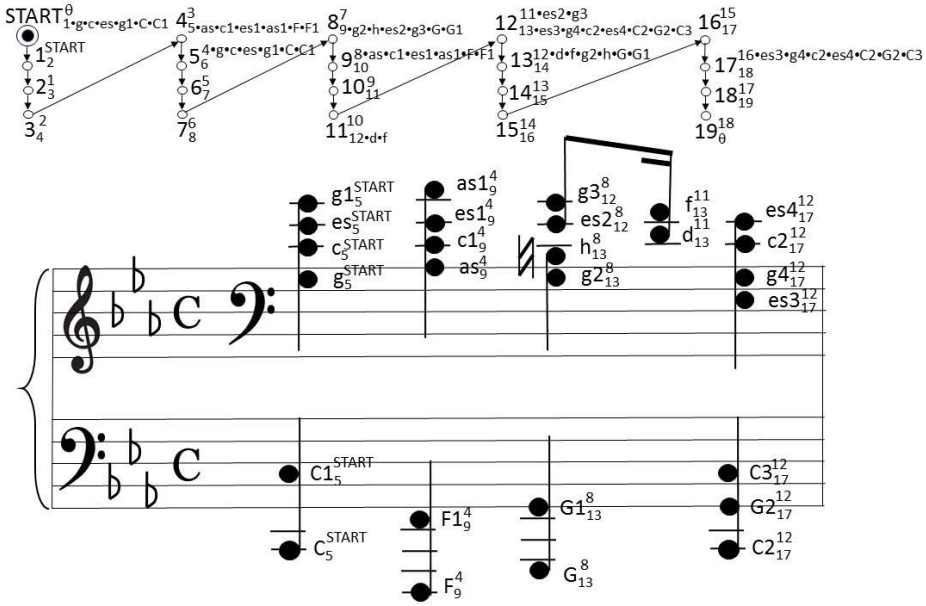


Fig. 10. First bar of the score of Prelude c-minor by Chopin, in the form of the min-time c-e-structure. The notes are represented as nodes with assigned duration periods, implemented by the control mechanism above the music text. The chords are accomplished by synchronization vertical notes, using multiplication "•"

The graphic examples have been tested by a computer program comprising editor and simulator of the cause-effect structures [Chm 2003].

A number of problems and properties of extended c-e structures, not presented in this short note, can be transferred from elementary c-e structures (see References). For instance such issues as:

- Decomposition of c-e structures
- Relation to Petri nets and to other models of concurrency
- C-e structures as lattices - their lattice properties
- Processes generated by c-e structures, monoid of processes
- Formal languages of c-e structure processes: the analysis and synthesis problems

Summarizing: the main motivation to develop the algebra (the quasi semiring) of c-e structures, was to combine structuring mechanism and transformation rules it provides, with appeal of simple pictorial and animated presentation of modelled real life systems. This algebra is a formal background for combining small c-e structures of easy to understand behaviour, into large system models, whose behavioral properties might be inferred from behaviour of their small parts. Such feature is called a *compositionality* (here conditional) - a counterpart of the *extensionality* in formal logic. Also, absence of explicit appearance of transitions and adjacent arrows - as is the case of Petri nets - provides more monitor space for graphic presentation.

References

[Chm 2003] Chmielewski R. *Symulacja struktur przyczynowo-skutkowych z wykorzystaniem platformy .NET* (in Polish), MSc thesis, Warsaw University 2003 (*Simulation of Cause-Effect Structures Using the .NET platform*)

[Cza 88a] Czaja L. *Cause-effect structures*, Information Processing Letters, 26, Jan.1988

[Cza 98a] Czaja L. *Cause-Effect Structures - Structural and Semantic Properties Revisited*, FUNDAMENTA INFORMATICAE 33 (1998) pp. 17-42, IOS Press, Amsterdam

[Cza 99] Czaja L. *Representing Hand-Shake Channel Communication in the Calculus of Cause-Effect Structures,* FUNDAMENTA INFORMATICAE, vol. 37, n. 4, March 1999, pp. 343-368

[Cza 2002] Czaja L. *Elementary Cause-Effect Structures*, Warsaw University, 2002

[Hol-Sza 88] Holenderski L., Szalas A. *Propositional Description of Finite Cause-Effect Structures*, Information Proc. Letters 27 (1988), pp. 111-117

[Mag-Mat 97] Maggiolo-Schettini A., Matteuci G. *Processes in Cause/Effect Systems*, FUNDAMENTA INFORMATICAE 31 (1997) pp. 305-335, IOS Press,

[Rac 93] Raczunas M. *Remarks on the equivalence of c-e structures and Petri nets*, Information Proc. Letters, 45 (1993) pp. 165-169

[Rei 85] Reisig W., *Petri Nets. An Introduction*, Number 4 in EATCS Monographs on Theoretical Computer Science, Springer, Berlin-Heidelberg-New York, Tokyo, 1985

[Ust 96] Ustimenko A.P. *Algebra of Two-level Cause-Effect Structures (revised version)*, Information Processing Letters, 59 (1996) 325-330

[Ust 98] Ustimenko A.P. *Coloured cause-effect structures*, Information Processing Letters, vol.68, No.5, December 1998, pp.219-225

[Wei-Gry 92] Weiss Z., Grygiel K., *Stochastic Cause-Effect Structures: A Simple Model*, Proc. of the Workshop "Concurrency, Specification and Programming", Berlin, Nov.1992

Automated Comparative Study of Some Generalized Rough Approximations
(Extended Abstract)

Adam Grabowski

Institute of Informatics, University of Białystok
Konstantego Ciołkowskiego 1M, 15-245 Białystok, Poland
adam@math.uwb.edu.pl

Abstract. The paper contains some remarks on building automated counterpart of the research presented during one of the previous *CS&P* workshops by A. Gomolińska. My main objective was the formal (and machine-checked) proof of Theorem 4.1 from her paper "A Comparative Study of Some Generalized Rough Approximations", hence the title of the present paper is by no means accidental.

Keywords: rough approximation, formalization of mathematics, Mizar Mathematical Library

1 Motivation

During *Concurrency Specification & Programming International Workshop* in 2001 Anna Gomolińska presented a draft of her paper on the generalization of rough approximation operators. Essentially, the idea behind the research was to collect some standard properties of rough approximations and to cut off some ordinary properties of binary relations to get – a little bit in the spirit of reverse mathematics – a catalogue of equivalences between relations' properties and corresponding formulas for rough sets. The final – revised and updated – version of the research appeared eventually in *Fundamenta Informaticae* under the title *A Comparative Study of Some Generalized Rough Approximations* [2] and contained also a brief review of various rough approximations, not necessarily classical ones.

During that time, I was also interested in computer-supported formalization of rough sets and I claimed that the paper could be a quite interesting testbed for my studies on the use of proof-assistants to support mathematicians in their work. Just to be a little more explicit, the system which was considered by me was Mizar, equipped with first order logic language and Tarski-Grothendieck set theory as a underlying set theory.

After a deeper study at that time, I decided to abandon the formalization work as it looked too much different from the one I followed as described in [5]. In the aproach chosen by Gomolińska, rough inclusion function κ and two uncertainty mappings – I and τ were quite fundamental at the very first first sight.

2 Approximations according to Gomolińska

Gomolińska started with – quite general – uncertainty map I from non-empty universe U into its powerset. Potentially, no additional assumptions at the beginning are made, with the exception of a kind of reflexivity – that any object u from the universe is an element of $I(u)$.

Theorem 4.1 is the core of the first part. We aimed at the full formalization of this item, so we quote it here using original notation from [2].

Theorem 4.1 For any sets $x, y \subseteq U$, objects $u, w \in U$, and $i = 0, 1$, it holds that:
a) $f_0^d \leq id \leq f_0$.
b) $f_1^d \leq id \leq f_1$.
c) $f_0(x)$ is definable.
d) $\forall u \in f_1(x).\kappa(I(u), x) > 0$.
e) $\forall u \in f_1^d(x).\kappa(I(u), x) = 1$.
f) If $\tau(u) = \tau(w)$, then $u \in f_0(x)$ iff $w \in f_0(x)$; and similarly for f_0^d.
g) If $I(u) = I(w)$, then $u \in f_1(x)$ iff $w \in f_1(x)$; and similarly for f_1^d.
h) $f_i(\emptyset) = \emptyset$ and $f_i(U) = U$; and similarly for f_i^d.
i) f_i and f_i^d are monotone.
j) $f_i(x \cup y) = f_i(x) \cup f_i(y)$.
k) $f_i^d(x \cup y) \supseteq f_i^d(x) \cup f_i^d(y)$.
l) $f_i(x \cap y) \subseteq f_i(x) \cap f_i(y)$.
m) $f_i^d(x \cap y) = f_i^d(x) \cap f_i^d(y)$.

The approach is significantly different than those of W. Zhu [11] (but Zhu's paper was published later), where the starting point is – at least in my opinion – more natural. Moreover, in the second part of the paper under discussion the catalogue of various rough approximation operators is given (which is outside Zhu's research area).

3 Mizar State of the Art

In our formal approach to rough approximations, our choice was to have indiscernibility relation ρ defined as a binary relation on a non-empty universe U. Essentially then, this corresponds to an abstract relational structure

$$\mathcal{R} = \langle U, \rho \rangle$$

with all properties credited to the internal relation of $\mathcal{R}$.

At the very beginning, we do not assume any of standard properties of approximations (or tolerances) added to the type of ρ, although we introduce two basic Mizar types, called `Approximation_Space` and `Tolerance_Space`, to have them available in the Mizar Mathematical Library. More thorough discussion on this development is given in [4].

Faithfully following Gomolińska, we should be aware of the fact that we have

$$\mathcal{A} = \langle U, I, \kappa \rangle$$

as a formal approximation space. I had some doubts if I should mention κ at all – rough inclusion function as in the classical approach the membership function was considered. Also ρ as indiscernibility relation was rather preferred over the uncertainty mapping. My convincement was stronger after the formalization of Zhu's paper on correspondence between properties of binary relations and of rough sets.

In the literature [11] the authors skip over the theory of less known properties of binary relations (or at least treat them as widely known), and I was also in a dead point as the – contrary to the notion of serial or mediate which could be claimed as mathematical folklore – positive or negative alliance relations were really unknown for me.

Even if in considered paper standard lower approximation operator is defined as (10)

$$LOW(x) = \{u \in U : \kappa(I(u), x) = 1\},$$

happily its immediate consequence is available as (12):

$$LOW(x) = \{u \in U : I(u) \subseteq x\}.$$

Then we could be sure the function κ will not be needed at the moment.

4 Generalizing Functions

Gomolińska claimed the following name space for approximations operators (18):

$$f_0(x) = \{u \in U : \tau(u) \cap x \neq \emptyset\}$$

$$f_1(x) = \{u \in U : I(u) \cap x \neq \emptyset\}$$

with corresponding f_0^d and f_1^d.

Then we can think on the following definition for τ

```
definition
  let R be non empty RelStr;
  let tau be Function of the carrier of R, bool the carrier of R;
  func f_0 R -> map of R means
    for x being Subset of R holds
      it.x = { u where u is Element of R : tau.u meets x };
end;
```

and similar, for uncertainty mapping I. Hence, it is quite straightforward to see that the common variable in both definitions is just arbitrary function (with proper domain and codomain), let us call it f, and then it is quite natural to use instead of the above formulation something like

```
definition
  let R be non empty RelStr;
  let f be Function of the carrier of R, bool the carrier of R;
  func ff_0 f -> map of R means
    for x being Subset of R holds
      it.x = { u where u is Element of R : f.u meets x };
end;
```

and then to use

```
definition
  let R be non empty RelStr;
  func f_0 R -> map of R equals
  ff_0 tau R;
  func f_1 R -> map of R equals
  ff_0 UncertaintyMap R;
end;
```

For a reader not very much acquainted with the Mizar system it is not very cryptic, but the use of a keyword `equals` turns automatic treatment of equalities on and then more reasonings can be justified automatically.

Furthermore, we have just a single – more general – theorem expressing (f) and (g) at the same time, instead of two:

```
theorem :: ROUGHS_5:31 :: 4.1 f) Flip
  for u,w being Element of R,
      x being Subset of R st
  f.u = f.w holds
    u in (Flip ff_0 f).x iff w in (Flip ff_0 f).x;
```

It can be noted at the first sight, that there is nothing on approximation operators there – we can think on quite general property of mappings as R is only non-empty relational structure.

The Mizar functor `Flip f`, which is the formal counterpart of f^d is defined accordingly as

```
definition let X be set;
           let f be Function of bool X, bool X;
  func Flip f -> Function of bool X, bool X means
    for x being Subset of X holds
      it.x = (f.x`)`;
end;
```

Obviously, no approximations are needed here, either.

5 Attributes

In our Mizar approach, UPP and LOW operators from [2] are classical upper and lower approximation operators, respectively. Theorem 3.1 from [2] lists them all, but it could be mentioned in this point, that all these – more or less standard – properties were already formulated and proved formally in Mizar script available in MML under identifier `ROUGHS_1` as a direct reflection of Pawlak's paper [7].

In our study of fuzzy implications, we developed some techniques (*Mizar schemes*) which could decrease the number of technical steps in defining functions.

It should be pointed out that if we start with the uncertainty mapping rather than with indiscernibility relation, the core property is

$$\forall_{u \in U}\ u \in f(u),$$

which is essentially the formula (1) in [2].

To have faithful formal representation is the above, we introduced the new Mizar attribute

```
definition :: property (1) p. 105
  let R be non empty set;
  let I be Function of R, bool R;
  attr I is map-reflexive means
:: ROUGHS_5:def 1
    for u being Element of R holds u in I.u;
end;
```

As we already mentioned before, our core idea was to start with an approximation space $\langle U, \rho \rangle$, where U is non-empty universe and ρ is an indiscernibility relation defined on U. All rough operators, membership and uncertainty functions, and also rough inclusions, could be defined based within this framework. The proper choice of the formal background is especially important, because in the Mizar Mathematical Library, all statements should be proven based on classical logic and/or already proven lemmas.

In this paper, as a rule we do not quote correctness conditions (as they are needed also for definitions), although definitions should also obey the rule of the necessary justification. Even if sometimes of quite technical character, the proof is needed and the example of that is given below. We formulate a bunch of handy *Mizar schemes* which make this activity a little bit less painful for the user.

```
definition
  let R be non empty set;
  func singleton R -> Function of R, bool R means
:: ROUGHS_5:def 2
  for x being Element of R holds it.x = {x};
  existence
  proof
```

```
    deffunc U(object) = {$1};
A1: now
      let x be Element of R;
      {x} c= R;
      hence U(x) in bool R;
    end;
    thus ex f being Function of R, bool R st
    for x being Element of R holds f.x = U(x)
      from FUNCT_2:sch 8(A1);
  end;
  uniqueness; ::: here the proof is not quoted
end;
```

Why this definition was really needed? Essentially, to use the Mizar type (and adjectives are important constructors for types) one should prove its non-emptiness, i.e., to construct at least one example of an object possessing this property, to show the usefulness of the attribute.

```
registration
  let R be non empty set;
  cluster singleton R -> map-reflexive;
end;
```

Another handy use of adjectives is given below:

```
registration
  let R, f;
  cluster ff_0 f -> c=-monotone;
end;
```

This allows to obtain automatically the proof of Theorem 4.1(i) as follows:

```
registration
  let R be non empty RelStr; :: i)
  cluster f_0 R -> c=-monotone;
  cluster f_1 R -> c=-monotone;
end;
```

6 Conclusions and Further Work

Obviously, building proper formal framework for further work is crucial. Having said that, we can build the rest of functions defined in [2]: $f_2, \ldots, f_9$, having ten mappings altogether. This allows for the possibility of automated reasoning on these operators, which could be made in the nearest future, as well as the reasoning on the – temporarily abandoned – rough inclusion function κ.

Even is this extended abstract is only a draft of the formal work done, the complete formalization (just accepted for inclusion into the Mizar Mathematical Library) is available at

`http://mizar.uwb.edu.pl/library/roughsets/`

Using 1386 lines of Mizar code (44 kilobytes) and formulating 7 definitions and proving 53 theorems we completed the first part of formalization work of Gomolińska's paper [2]. The `roughs_5.abs` file contains the abstract for the development while `roughs_5.miz` is the complete source file.

References

1. Bryniarski E.: Formal conception of rough sets, *Fundamenta Informaticae,* 27(2/3), 109–136 (1996)
2. Gomolińska A.: A comparative study of some generalized rough approximations, *Fundamenta Informaticae,* 51, 103–119 (2002)
3. Grabowski A.: Mechanizing complemented lattices within Mizar system. *Journal of Automated Reasoning*, 55:211–221 (2015)
4. Grabowski A.: Efficient rough set theory merging, *Fundamenta Informaticae,* 135(4), 371–385 (2014)
5. Grabowski A.: Automated discovery of properties of rough sets, *Fundamenta Informaticae,* 128(1–2), 65–79 (2013)
6. Järvinen J.: Lattice theory for rough sets, *Transactions on Rough Sets VI,* LNCS (LNAI) 4374, 400–498 (2007)
7. Pawlak, Z.: *Rough Sets: Theoretical Aspects of Reasoning about Data,* Kluwer, Dordrecht (1991). doi: 10.1007/978-94-011-3534-4
8. Rasiowa H. and Sikorski R.: *The Mathematics of Metamathematics*. Polish Scientific Publishers (1970)
9. Skowron A., Stepaniuk J.: Tolerance approximation spaces, *Fundamenta Informaticae,* 27(2–3), pp. 245–253 (1996). doi: 10.3233/FI-1996-272311
10. Wiedijk F.: Formal proof – getting started, *Notices of the American Mathematical Society,* 55(11), 1408–1414 (2008)
11. Zhu W.: Generalized rough sets based on relations, *Information Sciences,* 177(22), pp. 4997–5011 (2007). doi: 10.1016/j.ins.2007.05.037

The Bayes Theorem Counterpart in Mass-Based Rough Mereology

Lech Polkowski

Department of Mathematics and Computer Science. Chair of Mathematical Methods in Computer Science
University of Warmia and Mazury, Słoneczna str. 54, 10–710 Olsztyn, Poland
email: polkow@pjwstk.edu.pl

Abstract. In order to extend the formalism of the Bayes theorem to rough mereological universes, we endow the latter with a notion of a mass assigned to elements of the universe. We establish a counterpart of the Bayes theorem for mass endowed mereological universes in which rough inclusions are mass induced. We also establish an abstract form of the betweenness relation which has proved itself important in problems of data analysis and behavioral robotics. We point to applications in clustering and evidence theory.

Keywords: mass, rough inclusion, the Bayes theorem, betweenness.

1 Introduction

In our investigations into problems of data analysis and behavioral robotics as a model for intelligent agents, we have come at the theory of rough mereology as a useful milieu for these investigations. Rough mereology (or, as Achille Varzi has called it in Varzi [18] the fuzzified mereology) rests on the notion of a rough inclusion $\mu(x, y, r)$, a relation of being a part to a degree. In our study, we have applied some forms of rough inclusions, see, e.g., Polkowski [9],[12]. Basic forms of applied by us rough inclusions are linked to well-known idea of Pascal-Galileo of the relation of the number of favourable outcomes to the number of all outcomes of a random trial, the idea which in modern times was exploited, e.g., by Łukasiewicz [5] in assigning fractional truth values to indefinite formulae. We apply this idea of a fractional value towards a definition of an abstract class of rough inclusions based on a notion of a mass assigned to things in a considered universe. This idea allows us to relate for each pair x, y of things, degrees of partial containment of x in y and of y in x, a fortiori leading to a mass-based form of the Bayes theorem. The Bayes theorem known well from Probability Calculus, was investigated in the framework of rough set theory, cf. Pawlak [8], due to its ability of relating two-sided dependencies between pairs of things.

2 An outline of basics of mereology

We accept here the standard version of mereology as proposed in Lesniewski [4] in his pioneering work. The interested reader may as well consult, e.g., Casati and Varzi [2] or Polkowski [10]. Given some collection U of things regarded as individuals in ontological sense, a *relation of a part* is a binary relation π on U which is required to be

M1 *Irreflexive*: *For each thing* x, *it is not true that* $\pi(x, x)$.

M2 *Transitive*: *For each triple* x, y, z *of things, if* $\pi(x, y)$ *and* $\pi(y, z)$, *then* $\pi(x, z)$.

The relation of a part does induce the relation of an *ingredient ingr*, defined as

$$ingr(x, y) \Leftrightarrow \pi(x, y) \vee x = y, \tag{1}$$

which is clearly a partial order on things in U. The basic relation involving the notion of an ingredient is the relation of *overlapping*, $Ov(x, y)$ in symbols, defined as follows.

$$Ov(x, y) \Leftrightarrow \exists z.ingr(z, x) \wedge ingr(z, y). \tag{2}$$

The notion of overlapping in turn is instrumental in definition of the class operator in the sense of Leśniewski. This operator assigns to each non-empty collection of things F in the universe (U, π) its class, $ClsF$ which is a thing satisfying the two conditions:

(1) If $x \in F$ then $ingr(x, ClsF)$.

(2) If $ingr(x, ClsF)$ then for each y with $ingr(y, x)$ there exists $z \in F$ such that $Ov(y, z)$.

We are now in a position to recall here two *fusion operators* due to Tarski [16]. These operators are the *sum* $x+y$ and the *product* $x \cdot y$ defined by means of $x+y = Cls(z : ingr(z, x)$ *or* $ingr(z, y)$ and $x \cdot y = Cls(z : ingr(z, x)$ *and* $ingr(z, y)$.

The things x, y are *disjoint*, $dis(x, y)$ in symbols, whenever there is no thing z such that $ingr(z, x)$ and $ingr(z, y)$ (a fortiori, the product of x and y is not defined).

The *difference* $x - y$ is defined, when non-empty, as follows

$$x - y = Cls\{z \in U : ingr(z, x) \wedge \neg ingr(z, y)\}. \tag{3}$$

Rough (fuzzified) mereology is a theory of *rough inclusions*. Rough inclusions on a mereological universe U endowed with a part relation π are relations $\mu(x, y, r)$ on the product $U \times U \times [0, 1]$ cf. Polkowski and Skowron [15] and Polkowski [9],[10], [12].

They satisfy the following postulates, relative to a given part relation π and the induced by π relation *ingr* of an ingredient, on U:

RINC1 $\mu(x, y, 1) \Leftrightarrow ingr(x, y)$.

This postulate asserts that parts to degree of 1 are ingredients.

RINC2 $\mu(x, y, 1) \Rightarrow \forall z[\mu(z, x, r) \Rightarrow \mu(z, y, r)]$.

This postulate does express a feature of partial containment that a 'bigger' thing contains a given thing 'more' than a 'smaller' thing. It can be called a *monotonicity condition* for rough inclusions.

RINC3 $\mu(x, y, r) \wedge s < r \Rightarrow \mu(x, y, s)$.
This postulate specifies the meaning of the phrase 'a part to a degree at least of r'. From postulates RINC1-RINC3, and known properties of ingredients some consequences follow

1. $\mu(x, x, 1)$.
2. $\mu(x, y, 1) \wedge \mu(y, z, 1) \Rightarrow \mu(x, z, 1)$.
3. $\mu(x, y, 1) \wedge \mu(y, x, 1) \Leftrightarrow x = y$.
4. $x \neq y \Rightarrow \neg\mu(x, y, 1) \vee \neg\mu(y, x, 1)$.
5. $\forall z \forall r[\mu(z, x, r) \Leftrightarrow \mu(z, y, r)] \Rightarrow x = y$.

Property 5 may be regarded as an *extensionality postulate* in Rough Mereology.

3 Mass on a mereological universe

We introduce a new type of rough inclusions derived from a basic notion of a mass $m(x)$ assigned to each thing x in the mereological universe U endowed with a part relation π and the derived relation $ingr$ of an ingredient. The notion of mass in science is most often attributed to physical objects or linguistic category of mass expressions in dealing with which mereological tools are involved by some authors, cf. Nicolas [6]. Here, we introduce mass as an attribute of things which may admit various interpretations depending on the specific context of usage.

The notion of a mass m in what follows should satisfy the following demands:
MS1 $m(x)$ is positive real valued for each thing x in U.
MS2 If $ingr(x, y)$ then $m(x) \leq m(y)$.
MS3 If $dis(x, y)$ then $m(x + y) = m(x) + m(y)$.
Under those provisos, we define a rough inclusion candidate μ by letting

$$\mu(x, y, r) \Leftrightarrow \frac{m(x \cdot y)}{m(x)} \geq r. \tag{4}$$

Proposition 1. *The relation μ defined in (4) is a rough inclusion.*

Proof. Consider RINC1; assume $ingr(x, y)$ holds. Then $x \cdot y = x$, hence, $\frac{m(x \cdot y)}{m(x)} = 1$ and $\mu(x, y, 1)$. For the converse, assume that $\mu(x, y, 1)$ holds true. Was $\pi(x \cdot y, x)$, we would have $m(x - y) > 0$ and $m(x) = m(x \cdot y) + m(x - y)$, hence, $\frac{m(x \cdot y)}{m(x)} < 1$, a contradiction. It follows that $x \cdot y = x$ and thus $ingr(x, y)$. For RNC2, assume that $\mu(x, y, 1)$ and $\mu(z, x, r)$. Hence, $ingr(x, y)$ which implies that $ingr(z \cdot x, z \cdot y)$ thus $m(z \cdot y) \geq m(z \cdot x)$ and finally $\mu(z, y, r)$. RNC3 is obviously satisfied.

We notice that the rough inclusion μ is transitive:

$$\mu^+(x, y, \frac{m(x \cdot y)}{m(x)}) \wedge \mu^+(y, z, \frac{m(y \cdot z)}{m(y)}) \Rightarrow \mu(x, z, \frac{m(x \cdot y \cdot z)}{m(x)}). \tag{5}$$

Indeed, $m(x \cdot y \cdot z) \leq m(x \cdot z)$.

We now state an abstract mass-based form of the Bayes theorem.

Proposition 2. *(the Bayes theorem, simple form) (i)For any pair of things x, y we have: $m(x) \cdot \mu(x, y, r) = m(x \cdot y) = m(y) \cdot \mu(y, x, s)$ for appropriate values of $r \leq \frac{m(x \cdot y)}{m(x)}$ and $s \leq \frac{m(x \cdot y)}{m(y)}$, i.e., $\frac{r}{s} \cong \frac{m(x)}{m(y)}$.*

(ii) $\mu(x, y, r) = \frac{m(y) \cdot \mu(y, x, s)}{m(x)}$.

Remark 1. The form (i) as well as (ii) can be made more precise at the cost of introducing the modification of μ, denoted μ^+ which stands for the greatest value of the containment degree, i.e., $\frac{m(x \cdot y)}{m(x)}$.

$$\mu^+(x, y, \frac{m(x \cdot y)}{m(x)}). \tag{6}$$

Proposition 2 would then take the following form.

Proposition 3. *(a precisiated form of the Bayes theorem) For each pair x, y of things in the universe U, it holds that*

$\mu^+(x, y, \frac{m(x \cdot y)}{m(x)}) = \frac{m(y) \cdot \mu^+(y, x, \frac{m(x \cdot y)}{m(y)})}{m(x)}$.

The particular examples to the above general schema can be the following.

Example 1. Consider a finite universe U along with the collection of its non-empty subsets. For each subset x, let $m(x) = |x|$, i.e. the cardinality of x. In this case, the part relation is the relation of being a proper subset, the ingredient relation is the relation of being a subset, the sum $x + y$ is the union $x \cup y$, and, the product $x \cdot y$ is the intersection $x \cap y$.

Then $\mu(x, y, r)$ holds true if and only if $\frac{|x \cap y|}{|x|} \geq r$. The Bayes formula in Proposition 3 becomes

$$\mu^+(x, y, \frac{|x \cap y|}{|x|}) = \frac{|y| \cdot \mu^+(y, x, \frac{|x \cap y|}{|y|})}{|x|}. \tag{7}$$

Example 2. A parallel example is furnished by a collection of bounded measurable sets in a finite-dimensional Euclidean n-space, with the cardinality $|x|$ replaced with the n-volume $V^n(x)$.

Example 3. Consider a finite probability space (Ω, P) with the subspace $(\Omega^+, P|\Omega^+)$ of events with positive probability. In this case, the sum $x + y$ is the union $x \cup y$ and the product $x \cdot y$ is the intersection $x \cap y$. The mass $m(x)$ is now the probability $P(x)$ of the event x. The Bayes theorem 3 acquires now the form

$$\mu^+(x, y, \frac{P(x \cap y)}{P(x)}) = \frac{P(y) \cdot \mu^+(y, x, \frac{P(x \cap y)}{P(y)})}{P(x)}. \tag{8}$$

Example 4. Consider a collection of indefinite satisfiable formulae F over a finite universe U of things. For a formula f, the satisfiability set $Sat(f)$ is defined as $Sat(f) = \{u \in U : u \models f\}$. The rough inclusion μ is defined now as $\mu(f, g, r)$ if and only if $\frac{|Sat(f) \cap Sat(g)|}{Sat(f)|} \geq r$.

The Bayes formula in this case is like 7 with sets x replaced by sets $Sat(f)$. In case $\mu(f, g, 1)$ we say that the implication $f(x) \rightarrow g(x)$ is a true decision rule.

Example 5. Consider an information system $I = (U, A, V)$ with the universe of things U, attribute set A and the value set V: each $a \in A$ maps the set U into value set V and in consequence each thing $u \in U$ is represented as the set $I(u) = \{(a, a(u)) : a \in A\}$, cf., e.g., Pawlak [8].

We will consider the set $PI = \{(a, a(u)) : u \in U, a \in B, \emptyset \neq B \subseteq A\}$ of all partial non-empty sets of attribute-value pairs defined by things in U, with the ingredient relation defined by the subset relation. For a set $x \in PI$, we let $m(x) = |x|$.

Remark 2. We would like to mention the usage of the name 'mass' in Dempster-Shafer evidence theory cf. Dempster [3], where 'mass' is a substitute name for the 'basic probability assignment' m which denotes the values assigned to subsets of the frame of discernment Θ. In this case values assigned to those subsets sum up to 1 as they express degrees of belief that evidence is concentrated in a given subset. The belief function Bel defined for a subset Θ_i of Θ as

$$Bel(\Theta_1) = \sum \{m(\theta) : \theta \subseteq \Theta_1\} \tag{9}$$

satisfies the monotonicity condition for the our notion of a mass m but need not be additive.

4 A generalized Bayes theorem

Consider the mereological universe U. Let Y_1 be the maximal collection of things in U with the property:

(O) For each thing x in Y_1, x overlaps with the class $Cls(y_1 \setminus \{x\})$.

Let $y_1 = ClsY_1$.

Continuing in this way, we define y_2, y_3,.., y_n such that

(1) $U = y_1 + y_2 + ... + y_n$.

(2) $dis(y_i, y_j)$ when $i \neq j$ for $i, j + 1, 2, ..., n$.

We call the system $\{y_1, y_2, ..., y_n\}$ a *basis* for the mereological space (U, π, m).

For each $x \in U$, we have thus

$$x = x \cdot y_1 + x \cdot y_2 + ... + x \cdot y_n \tag{10}$$

and

$$m(x) = \sum_{i=1}^{n} m(x \cdot y_i) = \sum_{i=1}^{n} m(y_i) \cdot \mu^+(y_i, x). \tag{11}$$

The general form of the Bayes theorem can be stated now.

Proposition 4. *We have*

$$\mu^+(x,z) = \frac{m(x \cdot z)}{\sum_{i=1}^{n} m(y_i) \cdot \mu^+(y_i, x)}. \tag{12}$$

5 Relations to the notion of betweenness: a geometry for mass-based rough mereology

The notion of betweenness relation due to Tarski [17], modified by van Bentham [1] and adapted by us to the needs of data analysis and behavioral robotics will acquire here an abstract formulation in the framework of the mass mereology.

We introduce first the notion of distance $\delta(x,y)$ between two things x, y in the universe U. For $argmax_r \mu(x,y,r) = \frac{m(x \cdot y)}{m(x)}$ and $argmax_s \mu(y,x,s) = \frac{m(x \cdot y)}{m(y)}$, we let

$\delta(x,y) = max\{\frac{m(x \cdot y)}{m(x)}, \frac{m(x \cdot y)}{m(y)}\}$. Hence, $\delta(x,y) = m(x \cdot y) \cdot max\{\frac{1}{m(x)}, \frac{1}{m(y)}\} = m(x \cdot y) \cdot min\{m(x), m(y)\}$.

For a set $Y = \{y_1, y_2, ..., y_m\}$ of things in U, and a thing $x \in U$, we say that x is between things in Y, $B(x,Y)$ in symbols, in case the following condition holds true

$$B(x,Y) \Leftrightarrow \forall z. z = x \vee z \neq x \wedge \exists y \in Y. \delta(x,y) \geq \delta(z,y). \tag{13}$$

For a given $x \in U$ we denote with the symbol $BTW(x)$ the collection of sets $Y \subseteq U$ having the property that $B(x,Y)$.

Proposition 5. *The collection $BTW(x)$ is monotone, i.e., if $Y_1 \in BTW(x)$ and $Y_1 \subseteq Y_2$ then $Y_2 \in BTW(x)$.*

The following proposition sets some condition for betweenness.

Proposition 6. *Assume that there exists a subset $\emptyset \neq Y_0 \subseteq Y$ with the property that $ingr(y,x)$ holds true for each $y \in Y_0$. Then $B(x,Y)$ holds true.*

Proof. Consider $y_0 \in Y_0$. We have $y_0 \cdot x = y_0$. Hence, $\delta(x,y_0) = m(y_0) \cdot minm(x), m(y_0) = m^2(y_0)$. For $z \neq x$, we have

$\delta(z,y_0) = m(z \cdot y_0) \cdot min\{m(z), m(y_0)\} \leq m^2(y_0)$.

Remark 3. In particular cases, the betweenness relation can be described in more precise terms. In behavioral robotics, when mobile robots are modeled as planar rectangles, and the rough inclusion is defined as $\mu(A,B,r)$ if and only if $\frac{V^2(A \cap B)}{V^2(A)} \geq r$, it is shown that a robot A is between robots B and C if and only if A is contained in the minimal rectangle spanned on B and C as its diagonal vertices cf. Polkowski and Ośmiałowski [13], [14].

In the case of partial information sets in the set PI, we say that a set $x = \{(a, a(u)) : a \in B\}$ is a *convex combination* of sets $x_i = \{(a, a(u_i)) : a \in B_i\}$ for i=1,2,...,k where sets B_i are pairwise disjoint, $B = \bigcup_i B_i$, $(a, a(u)) = (a, a(u_i))$ for $a \in B_i$, $i = 1, 2..., k$. Then we prove that $B(u, \{u_i : i = 1, 2, ..., k\})$ holds true with respect to the rough inclusion $\mu(x,y,r)$ if and only if $\frac{|IND(x,y)|}{|x|} \geq r$, where $IND(x,y) = \{a \in A : a(x) = a(y)\}$ cf. Polkowski [11].

6 In search of an application: Clustering

Let us consider a possible mechanism for clustering based on mass rough inclusions. To this end, we introduce another distance function $\Delta(x, y)$ given by the formula

$$\Delta(x, y) = |\mu^+(x, y) - \mu^+(y, x)| = m(x \cdot y) \cdot |\frac{1}{m(x)} - \frac{1}{m(y)}|. \tag{14}$$

Given $\varepsilon > 0$, we consider the tolerance relation

$$\tau(x, y) \Leftrightarrow \Delta(x, y) \leq \varepsilon. \tag{15}$$

We define clusters as tolerance classes, i.e. maximal collections of things with the property that each pair of things in the collection are in the relation τ.

Let us provide a simple example.

Example 6. Consider things in a collection U being landscapes or photographs of a countryland on which we have trees, figures of people, houses. For a particular thing x we assign the mass $m(x)$ as the sum $m_1(x) + m_2(x) + m_3(x)$, where $m_1(x) =$ if and only if there are at least 3 trees on x, $m_2(x) = 1$ if and only if there are at least 2 people on x, and $m_3(x) = 1$ if and only if there is at least one house on x. We assume that $m(x)$ is at least 1 for each x in U. Figure 1 shows possible outcomes for pairs of things and values of ε for which these things may fall into one cluster. We have three possible types of things: Type I with m=3, Type II with m=2, and, Type III with m=1. We include into $x \cdot y$ a unit if and only if both x, y satisfy conditions for this unit, for instance if both x, y have $m_1 = 1$ then $m_1(x \cdot y) = 1$. We do not consider in this example the sum operation.

Table 1. Types of mass assignment towards clustering

Type x	*Type y*	$m(x \cdot y)$	$\Delta(x, y)$	ε *clustering*
I	*I*	3	$\Delta = 0$	*any positive*
I	*II*	2	$\Delta = \frac{1}{3}$	$\varepsilon \geq \frac{1}{3}$
I	*III*	1	$\Delta = \frac{2}{3}$	$\varepsilon \geq \frac{2}{3}$
II	*II*	2 *or* 1	$\Delta = 0$	*any positive*
II	*III*	0 *excluded or* 1	$\Delta = \frac{1}{2}$	
III	*III*	40 *excluded or* 1	$\Delta = 0$	*any positive*

It follows that for $\varepsilon < \frac{1}{3}$, clustering makes into clusters things of the same type: cluster 1 with things of Type I, cluster 2 with things of Type II, and, cluster 3 with things of Type III.

7 In search of an application: Making evidence approach decisive

In evidence theory (cf. Dempster, loc.cit.), mass assignments are also called basic probability assignments (b.p.a.'s) and they are assigned to all subsets of a set of possible outcomes called the frame of discernment. We illustrate our approach with an example.

Example 7. Imagine a car accident - a collision at the road crossing endowed with traffic lights. It is crucial to establish what light was on for the driver on the main road. Witnesses gave combined evidences resulting in the following b.p.a. m:

$m(red) = 0.25$,
$m(yellow) = 0.35$,
$m(green) = 0.20$,
$m(red\ or\ yellow) = 0.08$,
$m(red\ or\ green) = 0.02$,
$m(yellow\ or\ green) = 0.08$,
$m(red\ or\ yellow\ orgreen) = 0.02$.

From this assignment, values of the belief function, $Bel(A) = \sum_{B \subseteq A} m(B)$, are computed:

$Bel(red) = 0.25$,
$Bel(yellow) = 0.35$,
$Bel(green) = 0.20$,
$Bel(red\ or\ yellow) = 0.68$,
$Bel(red\ or\ green) = 0.47$,
$Bel(yellow\ or\ green) = 0.63$,
$Bel(red\ or\ yellow\ or\ green) = 1.0$.

We now compute values of rough inclusion taking as new masses for rough inclusions the computed values of Belief function. Hence, $\mu^+(x, y) = \frac{Bel(x \cap y)}{Bel(x)}$. These computed values of rough inclusions are collected in Figure 2. We omit the full set $\{r, y, g\}$ as the least decisive.

Table 2. Values of rough inclusions between sets of traffic lights

set	*red*	*yellow*	*green*	*red, yellow*	*red, green*	*yellow, green*
red	1.0	0.0	0.0	1.0	1.0	0.0
yellow	0.0	1.0	0.0	1.0	0.0	1.0
green	0.0	0.0	1.0	0.0	1.0	1.0
red, yellow	0.27	0.5	0.0	1.0	0.27	0.27
red, green	0.5	0.0	0.4	0.5	1.0	0.5
yellow, green	0.0	0.7	0.4	0.55	0.3	1.0

We introduce the measure of *independent evidence* $M(y)$ as the sum

$$\sum_{all\ non-singleton\ sets\ x \neq y} \mu^{+}(y, x). \tag{16}$$

These values are therefore : $M(red,\ yellow) = 0.54$, $M(red,\ green) = 1.0$, $M(yellow,\ green) = 0.85$. It follows that the maximally independent, having the smallest intersection/dependence on other sets is $red,\ yellow$. One decides that the light on the main road at the moment of crossing the crossroads was either red or yellow. Now in this set, the proportion of evidence for red to evidence for yellow is like 0.27:0.5 so the decision is on yellow light.

8 Conclusion

We have introduced the notion of a mass into rough mereology which has allowed us to express the reciprocal relations of partial containment in the form characteristical to the Bayes formula in probability theory. We have expressed the betweenness relation in an abstract mass-based . We proposed an application to clustering that allows for inducing various sets of clusters dependent on teh threshold distance ε.We hope that this abstract formulation will prove a convenient vehicle for some forms of approximate reasoning to be developed in future. At the end, we proposed a decision procedure involving mass based rough inclusions derived from belief values in evidence theory.

References

1. van Bentham, J.: The Logic of Time. Reidel. Dordrecht, 1983.
2. Casati, R., Varzi, A.C.: Parts and Places. The Structures of Spatial Representation. MIT Press, Cambridge MA, 1999.
3. Dempster,A. P.: Upper and lower probabilities induced by a multivalued mapping. Annals Math. Stat. 38, pp 325-339, 1967.
4. Leśniewski, S.: Foundations of the General Theory of Sets (in Polish). Moscow, 1916.
5. Łukasiewicz, J.: Die Logischen Grundlagen der Wahrscheinlichkeitsrechnung. Krakw, 1913. Cf. Borkowski, L. (ed.): Jan Łukasiewicz. Selected Works.North Holland-PWN, Amsterdam-Warszawa, pp. 16-63, 1970.
6. Nicolas, D.: The logic of mass expressions. In: Stanford Enc. Phil. Available: https://plato.stanford.edu/entries/logic-massexpress/.
7. Pawlak, Z.: Rough Sets: Theoetical Aspects of Reasoning about Data. Kluwer, Dordrecht, 1991.
8. Pawlak, Z.: A rough set view on Bayes's theorem. Intern. J. Intell. Systems 18(5), pp 478-498, 2003.
9. Polkowski, L.: Approximate Reasoning with Parts, An Introduction to Rough Mereology. Springer-Verlag, Berlin Heidelberg. ISRL vol. 20, 2011.
10. Polkowski, L.: Mereology in Engineering and Computer Science. In: Calosi, C., Graziani, P.: Mereology and the Sciences. Springer Synthese Library vol. 371, 217-292, 2015.

11. Polkowski, L.: From Leśniewski, Łukasiewicz, Tarski to Pawlak: Enriching Rough Set Based Data Analysis. A Retrospective Survey. Fundam. Inform. 154(1-4), pp 343-358, 2017.
12. Polkowski, L., Artiemjew, P.: Granular Computing in Decision Approximation. An Application of Rough Mereology. Springer Int. Publishers, Cham, Switzerland. ISRL vol. 77, 2015.
13. Polkowski, L., Ośmiałowski, P.: Spatial reasoning with applications to mobile robotics. In: Xing-Jiang, J. (ed.): Mobile Robots Motion Planning. New Challenges. I-Tech, Vienna, pp 433-453, 2008.
14. Polkowski, L., Ośmiałowski, P.: Navigation for mobile autonomous robots and their formations: An application of spatial reasoning induced from rough mereological geometry. In: Barrera, A. (ed.): Mobile Robots Navigation. InTech, Zagreb, pp 39-354, 2010. . I-Tech, Vienna, pp 433-453, 2008.
15. Polkowski, L., Skowron, A.: Rough mereology: A new paradigm for approximate reasoning. Int. J. Approx. Reasoning 15(4), pp333-365, 1997.
16. Tarski, A.: Zur Grundlegen der Booleschen Algebra I. Fund. Math. 24, pp177-198, 1935.
17. Tarski, A.: What is elementary geometry? In: Henkin L., Suppes P., Tarski A. (eds.): The Axiomatic Method with Special Reference to Geometry and Physics, North-Holland, Amsterdam, pp 16–29, 1959.
18. Varzi, A.: Mereology. In: Stanford Encyclopedia of Philosophy. Available: https://plato.stanford.edu/entries/mereology/

On some heuristic method for optimal database workload reconstruction

Marcin Zimniak[1], Marta Burzańska[2], and Bogdan Franczyk[1]

[1] Information Systems Institute
Leipzig University, Germany
{zimniak,franczyk}@wifa.uni-leipzig.de
[2] Faculty of Mathematics and Computer Science
Nicolaus Copernicus University, Toruń, Poland
quintria@mat.umk.pl

Abstract. The paper deals with the problem of database workload and its reconstruction. It is partially related to workload as a sequence of SQL statements in physical database design problem. An efficient algorithm based on greedy heuristic method for workload reconstruction using periodic patterns is provided. The quality of reconstruction is estimated by proposed reconstruction quality indicator.

Keywords: workload and workload reconstruction · periodic patterns· periodic patterns discovery· heuristic methods · optimization in physical database design .

1 Introduction

The problem of physical database design and tuning often requires detailed workload analysis. The paper "Automatic physical design tuning: workload as a sequence" [1] published in 2006 defines the structure of the workload as a sequence of SQL queries. This paper influenced a lot of research on workload, however, the topic of studying workload on an abstract plane to boost the performance of a database management system has not been fully addressed. The following paper introduces a new approach to the database workload based on the multiset concept. In this approach we do not analyze a sequence of SQL queries, instead, we take into account the multisets of queries abstract syntax trees (AST). Query syntax tree is viewed as the implementation of the SQL query under specific conditions at a specific time in a DBMS. Through the use of the AST concept, the problem of the equivalence of SQL queries in the process of generating the workload has been avoided. Similar problems have been discussed in previous papers [11, 12]. Both dealt with application of periodic patterns methods to a series of SQL queries. However, despite analyzing the workload on a physical level, both papers lack (among other things) the analysis of transition cost between queries and DBMS states. Such costs are important when working with bigger workloads and calculating their total cost, which then is used in various recommendation systems.

Prediction and reconstruction of the workload can become a useful tool for optimizing recommendation modules. At a later stage, they can be used in the development of some form of automated physical database design tools.
The following paper presents a new concept for the reconstruction of the workload. This concept is a result of combining of the data mining of periodic patterns with elements of physical database design. Those elements include cost models used in DBMS optimization. The aim of the article is to provide an effective heuristic method to search for the optimal workload reconstruction and also to provide the reconstruction quality measure. Both elements will be used in workload prediction and determination of the workload prediction degree.

This paper is organized as follows. The second section presents the concept of workload, including cost issues related to the physical workload model. The third section deals with defining periodic patterns and their derivation rules. This section also defines the workload reconstruction and reconstruction quality measure proposal. Section 4 introduces the algorithm which uses one of the heuristic methods in order to generate an optimal reconstruction. Section 5 concludes and discusses further research plans.

2 Workload

The article examines the workload on two essentially independent planes. On the abstract plane, we do not include cost relations between database objects and costs resulting from the transition from one database configuration to another. As for the physical workload model, it reflects the behavior of the database system for a given time period during which the aforementioned costs are taken into account.

2.1 Database processing model

We consider a typical relational database system where the relational model of data is used to represent data containers. Let x be a nonempty set of attribute names later on called as a *relational schema* and let $dom(a)$ denotes a domain of attribute $a \in x$. A *tuple* t defined over a schema x is a full mapping $t : x \to \bigcup_{a \in x} dom(a)$ and such that $\forall a \in x$, $t(a) \in dom(a)$. A *relational table* r created on a schema x is a set of tuples over a schema x.

Query processor transforms SQL statements submitted by the user applications into the query execution plans formulated as the expressions of extended relational algebra. The operations of extended relational algebra include the implementation dependent variants of operations of standard relational algebra such as *selection*, *projection*, *join*, *antijoin*, *set operations*, and other operations like *grouping*, *sorting*, and *aggregate functions*. Due to the different implementation techniques, the operations included in the basic system of relational algebra, e.g. *selection* or *join* contribute to an number of different elementary operations depending on their implementations, e.g. *index based selection*, *full scan selection*, *hash based join*, *index based join*, etc.

2.2 Abstract workload model

k SQL statements submitted by M users within the *user applications* $a_1, \ldots, a_n$ are recorded in an *application trace*. A *trace of an application* a_i is a finite sequence of pairs $<c_i{:}t_{c_i}, s_{i_1}{:}t_{i_1}, \ldots, s_{i_n}{:}t_{i_n}, d_i{:}t_{d_i}>$ where c_i is a *connect* statement, t_{c_i} is a timestamp when the statement has been processed, each s_{i_j} is SQL statement with a timestamps t_{i_j} attached, and d_i is a *disconnect* statement with its timestamp t_{d_i}. Processing of an application a_i starts from processing of a connect statement c_i, the processing of SQL statements s_{i_j}, and it finally ends with processing of a disconnect statement d_i.

An *audit trail* is a sequence of interleaved trails of user applications. For example, a sequence $<c_i{:}t_{c_i}, s_{i_1}{:}t_{i_1}, c_j{:}t_{c_j}, s_{j_1}{:}t_{j_1}, s_{i_2}{:}t_{i_2}, d_i{:}t_{d_i}\ d_j{:}t_{d_j}>$ is a sample audit trail from the processing of applications a_i, and a_j.

In the subsequent text the implementation record of each of the k SQL queries is placed within a non-empty period of time [a,b] comming from M users . It follows that syntactically equivalent `SELECT` queries can have different implementations. The problem of SQL query equivalence is a complex problem [3, 2].

We can circumvent this problem by using query execution plans accessible through the use of mentioned `EXPLAIN PLAN` command. Such plans usually take the form of enhanced syntax trees and are treated as query implementations. `SELECT` query analysis on a non-empty period of time [a,b] results in extraction of the syntax trees which are then placed in a *syntax tree table* [12]. This table contains a complete and compressed information about the syntax trees of SQL statements and the number of their occurrences in the analyzed workload. The paper [12] contains detailed information about the construction of such tables. It is worth noting, that a syntax tree is represented in a syntax tree table only once, no matter how many times it is included in the other syntax trees as a subtree. The cases of shared subtrees resulted in the adoption of the multisets theory. In the following text we define a multiset M is defined as a pair $<S, f>$ where S is a set of values and $f : S \to N^+$ is a function that determines multiplicity of each element in S and N^+ is a set of positive integers [9]. We also assume that the syntax trees have been unambiguously labeled by the letters of a fixed alphabet - a set of natural numbers.

For simplicity in further definitions, we assume the condition that the execution time of each query together with the generated load was recorded unambiguously for a given workload.

For the given time period $[a, b]$ and the number of analyzed SQL queries k, let $n \leq k$ be a minimal number of the time period's equal divisions such that the total execution time for each of the queries (including all implementational costs) fits in exactly one time segment with the length $|[a, b]/n|$. Such time period division generates n time segments of equal length called the *time units*. Each time unit has its established length and a start point in time [12].

Let U be a nonempty sequence of n disjoint time units over which a workload of k queries is recorded and let $|U| = n$ denote the total number of time units in U. Then $U[m]$ denotes the m-th time unit in U for $m = 1..n$. Let V be a mapping

of a subset $1..k$ of natural numbers representing workload queries (or more precisely query syntax trees) into a subset $1..n$ of natural numbers representing time units. This syntax tree-to-time unit mapping allows for registrations of syntax trees in syntax tree tables. Those tables are later used to locate similar syntax trees whose location in the [a,b] only slightly differs from the "ideal" periodicity.

Let L be a set of all syntax trees (including all syntax subtrees) generated from a given set of k SQL queries executed in a specified time period $[a, b]$ with a given sequence of time units U. For each $T \in L$, a *workload trace of a syntax tree* T is a multiset W_T of time units such that $W_T[i] = <\{T\}, f_i>$ and $f_i(T)$ is equal to the total number of times the syntax tree T was processed in the i-th time unit $U[i]$.

In addition, let the syntax tree table comprising all syntax trees and subtrees be given. A *workload* of the set L is denoted by W_L and $W_L = \biguplus_{T \in \mathbf{L}} W_T$

2.3 Physical workload model

In this paper, the aforementioned W_L structure was used instead of the earlier model of the physical workload considered in [1]. In addition, the following extension was adopted. Instead of the sequence of `SELECT` expressions, the sequences of multisets of syntax trees were used. Due to the use of the SQL query execution plans it was possible to register on-the-fly: operations, containers and access paths with costs (and workloads) at the level of each operation in the execution plan.

Let the enumeration of the syntax trees be monotonic through the set of natural numbers with accordance with the timestamp values. Let $\{S_k\}$ be a multiset of syntax trees with a given $U(t)$. Let $(S_1, S_2, ..., S_N)$ be a sequence of N multisets registered in W_L.

There are many methods for registering the `SELECT` queries in relational databases. An example of such a method in the Oracle DBMS is the so-called *audit trail* applied in [12]. In addition, there are built-in workload logging tools (eg. *Profiler* tool in Microsoft SQL Server). A *physical structures*hould be understood as any access path supported by the database server. Those structures include, among others: indexes, materialized views, multidimensional clustering of tables, etc. A *configuration* of the workload W_L is the set of possible-to-use physical database structures that can be materialized. A physical structure is considered *significant* if it can potentially be used in the execution plan of a `SELECT` query (even if it was not used in the final execution plan at the defined time period $[a, b]$). The topic of costs in database systems is a very broad subject, simplified in this paper. In order to have comprehensive knowledge about costs, eg. in the Oracle database system, we refer the reader to [6].

The following notation was used in the further part of the work. `COST (`$\{S\}$`,` C`)` means the total cost of operations in the `EXPLAIN PLAN` expression encoded with the appropriate syntactic trees at the given C database configuration. Let `TRANSITION-COST (`C_i`,` C_j`)` be the minimum cost of the transition between the C_i and C_j configurations. These costs include costs related to the creation /

removal of indexes and other physical structures. We assume the available optimization mechanisms that estimate costs on an ongoing basis, perform without unnecessary overhead using built-in extensions such as what-if, etc.

Representation of the $(\{S_1\}, \{S_2\}...\{S_N\})$ sequence execution is defined as a sequence (C_1, $\{S_1\}$, C_2, $\{S_2\}$ ... C_N, $\{S_N\}$, C_{N+1}). It is a sequence in which each multiset of syntax trees has a pre-configuration and post-configuration (we allow for empty configurations).

We define the *sequence execution cost* $<C_1, \{S_1\}, C_2, \{S_2\}...C_N, \{S_N\}, C_{N+1}>$, as $\sum_{k=1}^{N}$ (`COST` $(\{S_k\}, C_k)$ + `TRANSITION-COST` (C_{k-1}, C_k))+ `TRANSITION-COST` (C_N, C_{N+1}).
The zero state C_0 can be, for example, the initial state of the database or its value can be set by built-in what-if applications. All the costs discussed so far are accompanied by workloads and time units. In the further part of the article, we assume that in each $U(i)$, $i = 1, 2, ..., n$, the total workload is directly proportional to the total costs, treating the concepts of costs and workloads interchangeably.

3 Workload reconstruction using periodic pattern theory

Workload reconstruction plays an important role in the automated physical database design and in physical design optimization mechanisms. In this paper out of all possible reconstructions, we investigate only those most probable and, at the same time, the most acceptable when it comes to costs. It means we study those W_L into W_L mappings for which the total costs during reconstruction does not exceed initial total workload costs. Those mappings maintain the consistency of the subsequences implemented through a minimal set of periodic patterns with an emphasis on maximizing quality indicators of periodic patterns

3.1 Periodic patterns

The theory and applications of the concept of periodic patterns to the workload prediction problem were discussed in the previous works of one of the authors [11, 12]. The theory of periodic patterns is well known. It grew out of, among others, the periodic sets [7] as well as periodic events [8].

Let the workload W_L and the sequence of time units U be given. The sequences $C, C' \subseteq W_L$ of the same length are called *equivalent* if $C = C'$ occurs for all corresponding coordinates.

A *periodic pattern* in a workload W_L is a tuple $<C, f, t, p, >$ where:

1. the *carrier* C determines a non empty subsequence $C \subseteq W_L$
2. f is a number of time unit in U where the repetitions of C start
3. t is a total number of occurrences of equivalent sequences $C \subseteq W_L$, such that p denotes the number of consecutive time unit elements after which the t pairs of neighboring sequences are equivalent.
4. Parameters f, t, p satisfy the following inequality: $f, t \geq 1$, $p \geq 0$, $f + (t - 1) * p + |C| - 1 \leq |U|$

Also, if $t = 1$ then $p = 0$ and the pattern $<C,f,1,0>$ is called the *trivial periodic pattern* (*trivial* pattern)

Let $<C,\ f,\ t,\ p,>$ be a periodic patterns in W_L with a given U.
A *trace of a carrier* C is a subsequence $C \subseteq W_L$, denoted $tr(C, f, n)$, in which the first $f - 1$ elements are the empty multisets.
A *trace of a periodic pattern* $<C,\ f,\ t,\ p,>$ over the time unit sequence U, under the condition $f + (t-1) * p + |C| - 1 \leq n$, is a subsequence $TR(< C, f, t, p >, n)$ of a sequence W_L such, that $TR(< C, f, t, p >, n) = tr(C, f, n) \uplus tr(C, f + p, n) \uplus \ldots \uplus tr(C, f + (t-1) * p, n)$

3.2 Derivation rules

According to the work [4], for the periodic patterns we define the derivation rules by means of which new periodic patterns can be generated. Given the W_L and U, the following rules take place:
Rule 0 (Triviality) Let C be a submultiset, such that $C \subseteq W_L[f]$ for $f \in \{1, .., n\}$. Then $<C,f,1,0>$ is a (trivial) periodic pattern in W_L. This rule states that in any non-empty workload W_L , you can find all the trivial patterns of the form $<C,f,1,0>$.
Rule 1 (Normalization) Let $<C,f,t,p>$ be a periodic pattern in W_L. Then $<C',f',t,p>$, where $f' = f + i$, is a periodic pattern in W_L, such that C' is formed from C by the elimination of all of the i-empty multisets preceding C and/or the elimination of all of the empty multisets trailing C.
Rule 2 (Exclusion/Duality) Let $<C,f,t,p>$ be a periodic pattern in W_L. If $f_{split} = f + i * p$ for $0 \leq i \leq t - 1$, then only one of the following patterns is a periodic pattern: a) $P =<C,f,i-1,p>$ with $W_L = W_L \setminus TR(P', n)$ is a periodic pattern in W_L such that $P' = <\ C,\ f_{split},\ t - i + 1,\ p>$, b) $P' = <C,\ f_{split},\ t - i + 1,\ p>$ with $W_L = W_L \setminus TR(P, n)$ is a periodic pattern in W_L such that $P =<C,f,i - 1,p>$

Contrary to the previously mentioned research, in this paper we omit the concept of periodic patterns "validity". As a result, the process of building and applying derivation rules may result in "depletion" of the workload that takes place in the Exclusion/Duality rule
Rule 3 (Elimination) Let $<C,f_i,t_i,p_i>$, $<C,f_j,t_j,p_j>$ be periodic patterns in W_L, such that $f_i < f_j$. Then the following cases hold:

(1) If $t_i = t_j = 1$ then $\langle C,\ f_i,2,\ f_j - f_i\rangle$ cannot be a periodic pattern in W_L (the carrier C starting from position f_i can occur a maximum of 1 time in W_L - in accordance with the definition. The following sub-rules stating the maximum of t_i, t_j times starting from f_i, f_j respectively)
(2) If $t_i = 1$, $t_j > 1$ and $f_j - f_i = p_j$, then $\langle C,\ f_i,t_j + 1,\ p_j\rangle$ is not a periodic pattern in W_L.
(3) If $t_j = 1$, $t_i > 1$ and $f_j = f_i + t_i * p_i$, then $\langle C,\ f_i,t_i + 1,\ p_i\rangle$ is not a periodic pattern in W_L.
(4) If $t_j \neq 1$, $t_i \neq 1$, $p_i = p_j$ and $f_j = f_i + t_i * p_i$, then $\langle C,\ f_i,t_i + t_j,\ p_i\rangle$ is not a periodic pattern in W_L.

Rule 4 (Decomposition) Let $<C,f,t,p>$ be a periodic pattern in W_L. Then $\langle C',\ f,t,\ p\rangle$, where a carrier C' is a subsequence of a carrier C, is a periodic pattern in W_L.
Rule 5 (Composition) Let $<C_i,f_i,t,p>$, $<C_j,f_j,t,p>$ be periodic patterns in W_L, such that $f_i \leq f_j$ and $\uplus_{s\in\{i,j\}} TR(< C_s, f_s, t, p >, n) \subseteq W_L$. Then $\langle C_k,\ f_i,t,\ p\rangle$ is a periodic pattern in W_L such that $C_k = tr(C_i, 1, f_j - f_i + |C_j|) \uplus tr(C_j, f_j - f_i, f_j - f_i + |C_j|)$.

For example, given the periodic patterns $\langle TV^2,1,3,4\rangle$ and $\langle T,4,3,4\rangle$ in a workload W_L with given U then $\langle TV^2\emptyset T,\ 1,3,\ 4\rangle$ is a periodic pattern in W_L as well.

3.3 Reconstruction and workload reconstruction quality measure

The model theory, in George Polya's view, deals with the equivalence classes of similar periodic sequences. The motivation behind the reconstruction concept is the fact that for each sequence of determined processes (and with such we are working) there exists a period and pre-period [5]. The theory of shifts, in terms of periodic patterns for sequences, makes it possible to indicate the minimal sets of generators and their calculation is possible with the help of efficient algorithms. The problem raised in the work relates to parallel processes that interact with each other in real time. The study of the periodicity of such structures is close to the study of symbolic dynamics in particular of groups of automorphisms of similar structures.

Let R be a non-empty set of periodic pattens in a W_L given time unit sequence $U(n)$. We say that R is a *reconstruction of the workload* W_L in $U(n)$ if:

i. $\uplus_{s=1}^{|R|} TR(< C_s, f_s, t_s, p_s >, n) = W_L$
ii. all TRs implementing connect-disconnect processes remain consistent in relation to each other. We allow duplication of database connect/disconnect processes in case of hypothetical processes, assuming that logging in and logging out does not involve costs.

As a *quality measure of the reconstruction* R is a real value $0 \leq m_R < 1$ defined as:
$m_R = 1 - (1/\sum_{i=1}^{|R|}(\|C_i\| * t_i))^{1/|R|}$
where $\|C_i\|$ is the length of the carrier C_i, $|R|$ is the cardinality of R. When $R = R_0 = \{< W_L, 1, 1, 0 >\}$ we assume that $m_{R_0} = 0$.

Let the R_i, R_j be reconstructions in W_L with a given $U(n)$. We say that the reconstruction R_i is *better* (*more feasible*) than the reconstruction R_j (denoted $R_i > R_j$) if:

a) $m_{R_i} \geq m_{R_j}$,
b) $|R_i| \leq |R_j|$
c) the total sum of the sequence execution costs in R_i is not greater than the total sum of the sequence execution costs in R_j.
d) The number of the corresponding predictive patterns quality measures in the reconstruction R_i is greater than the respective number of measures in R_j, with at least one quality measure being taken into account.

The predictive patterns quality measures have been described in [4] and may easily be adapted to the generalized concept of predictive patterns described in [10].

There is one more qualitative measure of periodic patterns. Namely, the absolute number of different syntax trees in the C carrier of the given periodic pattern. If we have two different periodic patters P and P' generating the same costs, with P being more feasible then P' for most of the quality measures from [4], we say that P *dominates* over P' if $|supp(\{C_P\})| > |supp(\{C_{P'}\})|$ where $\{C_P\}$ is a multiset of the carrier C in a periodic pattern P.
The total cost of sequence execution in a reconstruction is the sum of the costs generated by the sequence of traces of all periodic patterns of the given reconstruction.

The concept of the database workload reconstruction presented in this paper was developed to predict the future database load. The benefits of estimating the optimal prediction are the databases optimization possibilities. Based on the database load forecast, one can create, for example, indexes, materialized views, etc. These structures can then be used at the right time in the future in such a way that, with their help, one can reconfigure significant structures even better than those proposed by existing advisory devices. Another possible application of the database workload reconstruction is the prediction of configuration. The encoding of `SELECT` queries using execution plan syntax trees enabled the current registration of important physical structures used in the query implementations. Using this fact at a further stage, it is possible to reconstruct the configuration in the given W_L, and thus to assess the quality of selection of physical structures used in the configurations.

The optimization issue for the reconstruction and thus for the estimation of optimal prediction is to determine the best reconstruction in the sense of the $>$ relationship described above.

Example 1. Let $W_L = < \{1\}, \{1^2\}, \{21^2\}, \{21\}, \{2\} >$. We may have a trivial reconstruction $R_0 = << \{1\}, \{1^2\}, \{21^2\}, \{21\}, \{2\} >, 1, 1, 0 >$ along with another cardinality 1 reconstruction $R_1 = << \{1\}, \{1\}, \{2\} >, 1, 3, 1 >$. Then $m_{R_0} = 0 < \frac{8}{9} = m_{R_1}$, wherein the traces of the reconstructions R_0 and R_1 are identical and thus the total costs of the sequence execution overlap in both reconstructions. From this it follows that R_1 is better than R_0 reconstruction.

Example 2. Let $W_L = < \{1\}, \{1^2\}, \{21^2\}, \{21\}, \{2\} >$ be a workload with a given $U(n)$ and a pair of indexes I_1 and I_2 used in the implementation of certain `SELECT` queries stored as syntax trees 1 and 2. The coded information in the syntactic tree is, among others, the access path, i.e. a complete set of relevant physical structures used in implementations. In this example it means that in the trees 1 and 2 the indexes I_1, I_2 were used respectively. In addition, we assume that the database storage is not affected in any time unit $U(i)$. Assume that the costs of creating indexes I_1, I_2 are the same regardless of where they are created. In addition, assume that based on the value of costs stored in the syntactic tree table, the costs of implementing the syntactic trees 1, 2 are respectively 3: 4. The deviation of this ratio does not exceed 10% if the expressions are

performed together in the same $U(i)$. In the case when the syntax trees 1 and 2 are executed separately (ie in different $U(i), U(j)$) then the cost of the syntax tree 1 is twice as high as the cost of the syntax tree 2 with a deviation not exceeding 5%. The benefits of using both indexes are the same for both syntactic trees regardless of whether 1 or 2 are executed together or separately. In addition, the costs of removing indexes I_1 and I_2 are equal 0. The calculations of the total reconstruction sequence costs show that the sum of the execution costs of the periodic patterns sequence in $R_1 = \{<< \{1\}, \{1\}, \{2\} >, 1, 3, 1 >\}$ is lesser than a respective sum in $R_2 = \{<< \{1\} >, 1, 3, 1 >\}$ $\{<< \{1\}, \{2\} >, 2, 3, 1 >\}$. Moreover the conditions a) $m_{R_0} = \frac{8}{9} > \frac{2}{3} = m_{R_1}$ with b) bring that $R_1 > R_2$.

4 Greedy heuristic method for workload reconstruction

The goal of the presented heuristic is to find the optimal, in the sense of the aforementioned relation $>$, reconstruction R in a set of all reconstructions for a given W_L workload (not exceeding actual costs) at the given $U(n)$. In the algorithm, the input data is: W_L in the form of sequences of natural numbers multisets, generally understood implementation costs of all of the syntax trees registered in W_L, as well as the transition costs between individual neighboring configurations. It is further assumed that the W_L coding is given by a sequence of multisets of natural numbers.

The motivation for the heuristic algorithm adaptation for configurations for W_L heuristic reconstruction was the observation that in the physical workload structure, each multiset is inextricably linked to a certain configuration. As shown by the numerous tests in the paper [1], heuristic solutions for configurations are sub-optimal. It can, therefore, be expected that the heuristics for reconstruction will proceed in the same way. Below we present a heuristic algorithm for reconstruction.

The Algorithm

1. Let $S = \{s_1, .., s_M\}$ be a set of physical structures in a given workload W_L. Using the exhaustive method to find the shortest path in the cost edge graph [1], a set of optimal solutions P for each of the s_i is calculated separately. As a result, we get a set $P = \{p_1, .., p_M\}$. Let $p_i =< a_{i_1}, W_L[1], .., W_L[n], a_{i_{N+1}} >$.

1.1. Let $R := \emptyset$, `while` ($W_L <> \{\emptyset\}$) `do:`
`for` i: = 1 to n `do:`
`for each` $j \in$ supp($\{W_L\}$) `do:`
1.1.1. $< W_L[i..n], i, 1, 0 >:=< W_L[i..n] \setminus \{st^i_j\}, i, 1, 0 > \cup < \{st^i_j\}, i, 1, 0 >$,
$< W_L[1..n+1-i], i, 1, 0 >:=< W_L[1..n+1-i] \setminus \{st^{n+1-i}_j\}, n+1-i, 1, 0 >$
$\cup < \{st^{n+1-i}_j\}, n+1-i, 1, 0 >$ where st^i_j is j-support element at $W_L[i]$ and $W_L[i..n] :=< W_L[i], W_L[i+1], .., W_L[n] >$.

In each 1.1.1 execute as follows: $R_i := \emptyset$. For each (pairwise) disjoint sequences (represented by the traces of the corresponding trivial periodic patterns), use (for individual sequences (traces) respectively): decomposition rule which is a preserving cost-based pruning technique and then apply any of the rules of: composition and/or exclusion and/or elimination. Proceed in such a way

that in the final result of this step you get a minimal set of periodic patterns R_i with the minimum total value of the sequence execution costs. This set takes into account the optimal "path" of the solutions given by the current p_i and the maximum value of the quality reconstruction measure m_{R_i}.
2. Let C be the set of all configurations over the p_i elements.
3. Greedy heuristics for R runs as follows:

3.1. Let $r=< c_1, W_L[i], ..., c_N, W_L[n], c_{N+1} >$ be the best configuration in P in terms of the total costs. Let rr be the best reconstruction in terms of a quality measure in R and such that its cost is the closest to the cost of configuration r. Then $P := P \setminus \{r\}$, $R := R \setminus \{rr\}$. Let $C := C \cup \{c_1, ..., c_{N+1}\}$.

3.2. Select the element s from the set P such that $t =$ `UnionPair(r,s)` (defined in [1]) is a configuration such that its value in terms of the total sequence execution costs among all P configuration is the smallest. Similarly, find in R a reconstruction R_s which is the smallest in R in terms of total costs. In addition, the cost of executing the sequence for the configuration t is smaller than the corresponding costs for the configuration r. Parallel, find in R such a reconstruction R_t which in terms of total costs is closest to configuration t (costs are not greater than t). If there is no s element, then proceed to step 4. Assign $P := P \setminus \{s\}$, $P :=P \cup \{t\}$, $R := R \setminus \{R_s\}$, $R :=R \cup \{R_t\}$. Go to step 3.1.

4. Create a graph for all configurations with C at each level (for every support-element in every $W_L[i]$). Find the shortest path in this graph. From the set R, return the reconstruction that corresponds to the shortest path in this graph.

5 Conclusions and future work

The paper presents a new concept of the workload reconstruction along with a measure of reconstruction quality and an efficient algorithm for generating optimal workload reconstruction, thus estimating the workload prediction quality. Unlike previous periodic pattern detection techniques based on the top-down methodology, the presented recursive approach accelerates the periodic pattern detection algorithm through a different derivation system. This system is mostly based on the reducing derivation rules. This results in reduction of the workload elements that need to be analyzed, which influences the speed of workload analysis.

The search for new heuristics, comparative tests, accuracy, and testing the properties of the proposed measure of quality is the next stage of research. Searching for the proposals for other quality measures and thus new criteria for reconstruction is also included in that stage. In order to verify the efficiency and quality of the presented algorithm, an implementation based on a "live" load course is planned. However, acquiring real companies strategic data is extremely difficult. Currently, the development phase includes an extended implementation including cost optimization. Lastly, more extensive research on workloads containing SQL-99 recursive queries should be conducted.

References

1. Agrawal, S., Chu, E., Narasayya, V.R.: Automatic physical design tuning: workload as a sequence. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. pp. 683–694 (2006)
2. von Bültzingsloewen, G.: Optimierung von sql-anfragen für parallele bearbeitung. In: Grundlagen von Datenbanken. pp. 20–22 (1990)
3. Ceri, S., Gottlob, G.: Translating sql into relational algebra: Optimization, semantics, and equivalence of sql queries. IEEE Transactions on software engineering **11**(4), 324–345 (1985)
4. Getta, J.R., Zimniak, M., Benn, W.: Mining periodic patterns from nested event logs. In: Proceedings of the 2014 IEEE International Conference on Computer and Information Technology. pp. 160–167. IEEE Computer Society (2014)
5. Goles, E., Martínez, S.: Neural and automata networks: dynamical behavior and applications, vol. 58. Springer Science & Business Media (2013)
6. Lewis, J.: Cost-Based Oracle Fundamentals. Springer (India) Pvt. Limited (2006), `https://books.google.pl/books?id=85iWawYUsVsC`
7. Matos, A.B.: Periodic sets of integers. Theoretical Computer Science **127**(2), 287–312 (1994)
8. Serafini, P., Ukovich, W.: A mathematical model for periodic scheduling problems. SIAM Journal on Discrete Mathematics **2**(4), 550–581 (1989)
9. Simovici, D.A., Djeraba, C.: Mathematical tools for data mining : set theory, partial orders, combinatorics. Advanced information and knowledge processing, Springer, London (2008), `http://opac.inria.fr/record=b1133711`
10. Zimniak, M., Getta, J.R.: On systematic approach to discovering periodic patterns in event logs. In: International Conference on Computational Collective Intelligence. pp. 249–259. Springer (2016)
11. Zimniak, M., Getta, J.R., Benn, W.: Deriving composite periodic patterns from database audit trails. In: Asian Conference on Intelligent Information and Database Systems. pp. 310–321. Springer (2014)
12. Zimniak, M., Getta, J.R., Benn, W.: Predicting database workloads through mining periodic patterns in database audit trails. Vietnam Journal of Computer Science **2**(4), 201–211 (2015)

Automatic validation of big data classifiers on multiple di verse datasets

Automated testing of big data classifiers

Przemysław Czaus

Department of Mathematical Methods of Computer Science, Faculty of Mathematics and Computer Science, University of Warmia and Mazury in Olsztyn, Słoneczna 54, 10-710 Olsztyn, Poland

czaus@matman.uwm.edu.pl

Abstract. While working on data mining applications the main questions are: what do we want to know based on the given data and is the result worth the additional computing power designated for the task. Given the diversity of data and implementations, it is important to select the best-optimized solution for the given task. Every algorithm can behave better or worse when implemented in different languages or even deployed on different architectures. With the expansion of cloud services, distributed programming solutions and containers, optimization even on system level is possible with less effort. The main problem is knowing if the selected solution is better than what was used before. Having the possibility of optimizing the system, algorithm, implementing the solution in a different language or even cleaning the data a different way may give a significant advantage. Most of the published results compare two similar algorithms, on a single machine, written in the same language. The tests differ between scientific manuscripts, sometimes using the same datasets, but without providing the resulting cleansed dataset. That makes the context of the results very narrow and hard to interpret in a bigger scope. The root problem is running unified tests on a variety of solutions and optimization.

Allocation of more resources for the same task sometimes isn't possible and can lead to data loss. Given the scale of some datasets, it is more practical to know if the changes are economically justified. That makes testing some changes on production environments difficult or even impossible. When selecting classifiers, one must first run his own tests using datasets, the same or similar to the production data. Simplifying the process shortens the time from an idea to selecting the best solution for the given job. My main focus is to ensure that every solution is being tested with regard to all of the most important parameters. This way we can measure the impact of changes in the same algorithm as well as the differences between classifiers using the same datasets. Giving a mechanism for standardized tests of new cleansing algorithms, classifiers, language implementations may result in a dynamic progress in a field of data mining. This way one

can find the best solution and the main differences in a matter of minutes depending on the computing parameters defined for the system.

The main purpose of our study is to build an automated system based on a distributed architecture. Instead of testing the solutions on a single local machine we use a cluster of machines. Those machines can have different hardware. Comparing test results from a base machine with a machine added to the cluster may allow to calculate an accurate difference in processing power so running the same tests on machines with the same architecture shouldn't be needed. For now, the cluster is built on Rancher, that allows deployment of new versions in a matter of seconds. The application consists of loosely coupled modules for dataset storage and cleansing, classifier storage, test generator and automated test runner. Datasets are stored with all their cleansed and test versions. That allows us to monitor the differences between the generated datasets and the result of filtering the data. Tests are generated based on a single cleansed dataset, this way we can see how one data cleaning algorithm impacts the tested implementation. Classifiers are stored in containers, and saved in a container database. The main focus is to ensure every container implements the basic interfaces for communicating with the application (for learning, visualisation and validation). The application communicates with containers asynchronously sending a job to the container and waiting for a response on a designated endpoint. Every classifier has its container version, and new versions are being generated with the given dataset. This way we don't need to teach the classifier when changing any of the system parameters or the processor architecture. Using the container ecosystem gives us the possibility to set some of the system restrictions like a number of processor cores, the size of the memory or even control the number of containers running at the same time. All the containers used for tests are being run in a dedicated cluster. This gives control over the load for the whole system and makes the results more reliable. Adding new architectures and servers to the system is easy and can be done while the application is running. If the whole architecture isn't used some of the servers can be turned off to ensure low maintenance costs of the architecture.

For an example, we want to add a new classifier to the database. We want to test the classifier using already defined standardized tests. First, we have to prepare a container containing a REST API that is implementing basic interfaces used by the application and the classifier that those interfaces send data to. We define what tests to run and we add this job to the queue. If some jobs are currently running we must wait for the processes to finish. The first step is to start the containers and try to teach them with the datasets. If everything is finished we store a snapshot of the container with the data loaded into the database. Every container that finished this process is terminated and the application waits for every container to finish. The next step is to run tests on the previously prepared containers. Every cluster has it's calculated limits and we can run as many copies of the apps at the same time as far as we don't exceed the limit. Every test has its own dataset for learning and for validation. To start the tests we send a package with validation data, this way the test is run locally and we don't have any network delays during the tests. The container measures the time it started

and begins to run the tests. When it finishes it sends the results back to the server, where it is stored and prepared for analysis and publication.

For now, the application can store multiple datasets and their versions with regard to the used cleaning algorithms. One can define default automated tests for new classifier implementations. An advantage for active development is a possibility to build a graph of versions, allowing to analyze what changes generated better results at different datasets. Additionally, we can queue our tests and check the results after everything has been generated.
This solution may be a great way to unify the testing of new classifiers or any algorithms working on cleaning the datasets. Giving everybody a way for fast validation of results of their work. Publishing test results may additionally help many people choose the best solution for a certain task.

Keywords: big data, datasets, classifiers, automated tests

Investigating Characteristics and Differences Between Easy and Hard SAT Instances (Extended Abstract)

Teofil Sidoruk[1]

Institute of Computer Science PAS, Warsaw
t.sidoruk@ipipan.waw.pl

Abstract. Following our comparison of the efficiency of SAT-solvers [19, 20], we analyse DIMACS input files previously generated for benchmarking purposes in an attempt to pinpoint some common characteristics for the CNF formulas that were relatively easier to process, i.e., were verified faster than comparable instances of the same size.

1 Introduction

Since the early 1970s, when Cooke first proved it to be NP-complete, the Boolean satisfiability problem, or SAT, has undergone a dramatic rise in importance. From the subject of purely theoretical research in the area of computational complexity, SAT-solving algorithms have become the cornerstone of a broad range of important practical applications that rely on their efficiency. They include, but are not limited to: verification [1, 2], (un)bounded model checking [5, 7, 13, 24, 25], planning [15], and composition of web services [18]. It is equally important to note that the theoretical aspects of SAT also remain the subject of keen scientific interest.

In our recent papers [19–21], we presented notable SAT-solvers, both state-of-the-art and historical, comparing their efficiency at several computational problems of varying complexity: from P-complete chess problems to EXPTIME-complete Towers of Hanoi puzzle. One obvious observation stemming from our comparison is that no single SAT-solver is superior to others in the sense that it always performs faster regardless of the input. In other words, solving SAT remains a considerable challenge: despite the incredible progress made, especially in the last fifteen years, the potential for further improvement is as large as ever.

The focus of this paper is not on SAT-solvers as such, but rather, on the input Boolean formulas themselves. Specifically, we will investigate CNF formulas that are comparatively easier or harder to verify compared to other generated instances of the same size, attempting to identify some common patterns in their properties. Given that this area that has not really been previously explored, this work is aiming to be an initial, small step rather than an exhaustive investigation.

The rest of this paper is organized as follows. The next section summarises existing work related to the subject. Section 3 shortly presents DIMACS, the standardised input format used by SAT-solvers, as well as details the generation and analysis of input CNF formulas. In Section 4 experimental results are compared and discussed. The final section contains conclusions.

2 Theoretical Overview and Related Work

In this section we discuss prior research into the difficulty of randomly generated instances of NP-complete problems.

Many well-known, classical computational problems, though NP-complete, are relatively easy to solve when it comes to typical instances [6]. The graph k-coloring problem, for example, was found to be solvable in logarithmic time in the vast majority of cases [23]. On the other hand, since their complexity was proven in Karp's seminal 1972 paper [14], we are bound to encounter hard instances eventually. This brings about the question of whether there is any pattern to be found in the distribution of difficulty in a set of randomly generated instances, which has been the subject of research since at least the early 1990s. In the rest of this paper, we will focus on the Boolean satisfiability problem (SAT), since it serves as the convenient 'common denominator' to which other hard problems are often translated.

It has been long observed that certain specific instances of SAT pose an unusually significant challenge to the the DPLL algorithm, contrary to perceived average difficulty. In [6] Cheeseman, Kanefsky and Taylor summarise classical NP-complete problems using 'order parameters'. For example, a set of instances of the Hamiltonian path problem can be ordered by the average connectivity of their respective graphs: the higher the connectivity, the higher the chance for a Hamiltonian path to exist. Furthermore, the authors show the existence of a phase transition at the boundary marked by some critical value of the order parameter, which separates two distinct regions of likely satisfiable and likely unsatisfiable instances, both of which are comparatively easy to verify. It is at the boundary that the hardest instances occur.

This phase transition is investigated further by Gent and Walsh in [11]. Their experimental results confirm the association of hardest instances of problems with the boundary, and that median problem difficulty generally follows the expected easy-hard-easy pattern. However, they also show that the distribution of difficulty is significantly more complex, and in particular note the presence of a region where instances can be extraordinarily difficult, sometimes orders of magnitude harder than those closest to the phase transition.

Gent and Walsh postulate the 'constraint gap' to cause such unexpectedly hard problems to occur in an otherwise satisfiable region. In the DPLL algorithm, neither unit propagation nor pure literal elimination ever branch out the search, leaving splitting (i.e., the choice of the branching literal) as the only critical point which can potentially result in an exponential blow-up in the number of explored assignments. This naturally leads to the conclusion that the harder the instance, the more the algorithm is forced to use the splitting rule compared to the other two. In other words, the hardest instances are 'constrained' in the sense they have just enough constraints to be unsatisfiable, but very few more (or even none), forcing DPLL to utilize heuristics-based branching and thus increasing verification time dramatically.

These results were further experimentally confirmed in other papers, including analyses for 3-SAT formulas by Larrabee and Tsuji [16] and by Crawford and Auton [8], with the latter work focusing on how the percentage of satisfiable instances changes as a function of the clause/variable ratio of the formula.

3 The Input Format, Formulas and Analysis

In this section we present the DIMACS input format commonly used by SAT-solvers, discuss the input files used for our analysis and the factors taken into account during the latter.

The renewed scientific interest in the Boolean satisfiability problem, and in particular the emergence of SAT Competitions in the early 2000s, resulted in the need of a single, unified input file format. DIMACS has become such a standard.

The format uses plain text to represent a Boolean formula in conjunctive normal form (CNF). Following an optional comment line and a header containing the number of clauses and literals in the formula, each subsequent line corresponds to a new clause. Variables are represented by subsequent natural numbers, with the minus sign denoting negation. Spaces separate literals in clauses, and zeroes signal end of clause. An example of a very simple CNF formula in the DIMACS format is shown below.

Listing 1.1. The input file corresponding to a simple formula $(x1 \vee \neg x3) \wedge (x2 \vee x3 \vee \neg x1)$

```
c Example DIMACS input
p cnf 3 2
1 -3 0
2 3 -1 0
```

The input files contain formulas resulting from the translations to SAT of several NP-complete problems, including classical graph problems (vertex coloring, vertex cover, Hamiltonian path) [14], as well as the extended string-to-string correction problem (ESCP). They were originally created as benchmarks for our previous work, i.e., a comparison of SAT-solvers [19, 20].

For the purposes of this analysis, we have identified and separated groups of the most and least difficult instances for each of the aforementioned problems, that is, the input files whose processing required the most and the least time, respectively. When calculating verification time, the average of individual solvers' processing times was considered. The solvers used were the same as in the aforementioned comparison: Lingeling and Plingeling [4], Glucose and Glucose-syrup [3], Clasp [10], Minisat [22], ManySAT [12] and Microsoft Z3 [9]. However, zChaff [17] was excluded due to its age and inability to process many instances in reasonable time, which would have considerably skewed the average.

The DIMACS files in both groups were subsequently analysed and compared w.r.t. factors such as total number of literals and clauses, average and maximum clause length, percentage of negative literals and percentage of Horn clauses.

4 Results

In this section we discuss and compare the results of our analysis.

In Tab. 1, several characteristics are compared between the groups of easiest and hardest generated instances of vertex k-colouring and vertex k-cover. For the former, parameters of $n = 100$ (graph size) and $k = 10$ (number of colours) were set. For the latter, graphs of size $n = 50$ were generated, with the vertex cover size at $k = 30$.

	Vertex k-colouring		Vertex k-cover	
	Easier instances	Harder instances	Easier instances	Harder instances
Avg running time	0.018 s	364.759 s	0.108 s	38.941 s
Avg number of clauses	24507	36007	36774	36994
Avg number of literals	49813	72813	74967	88127
Avg clause size	2.033	2.022	2.039	2.382
Longest clause	10	10	60	60
Negative literals	97.96%	98.63%	98.04%	83.41%
Horn clauses	0%	0%	99.03%	99.34%

Table 1. Comparison of characteristics between easier and harder instances of the vertex k-colouring and k-cover problems.

	Hamiltonian path		String correction	
	Easier instances	Harder instances	Easier instances	Harder instances
Avg running time	4.810 s	34.101 s	0.538 s	42.968 s
Avg number of clauses	8000200	8000200	49808	49808
Avg number of literals	20135132	17115771	318476	318476
Avg clause size	2.517	2.139	6.394	6.394
Longest clause	200	200	66	66
Negative literals	80.66%	93.25%	19.67%	19.67%
Horn clauses	0%	0%	0%	0.01%

Table 2. Comparison of characteristics between easier and harder instances of the Hamiltonian path problem and ESCP.

It can be observed that the characteristics of hard instances depend primarily on the computational problem and its specific translation to SAT. For instance, in the vertex k-colouring problem, they have up to 50% more literals and clauses. This in turn can be attributed to the randomly generated input graphs for these instances having significantly more edges, and as such requiring more constraints in the form of clauses. Notably, despite the overwhelming majority of literals in both groups of instances being negative, there are no Horn clauses, again due to the specifics of the SAT encoding.

On the other hand, for vertex k-cover the average number of clauses is roughly the same in easy and hard instances. However, the number of literals, and thus average clause length, is generally higher (up to 15%) in the latter group. Furthermore, an even more noteworthy difference is in the percentage of negative literals, which is also around 15% lower, suggesting that the extra literals in hard instances are positive.

In the same way, Tab. 2 compares instances of the Hamiltonian path problem and the extended string-to-string correction problem (ESCP). For the former, generated graphs were of size $n = 200$, whereas the parameters for ESCP were set as $n = 20$ (length of input strings), $k = 15$ (maximum number of operations) and $l = 5$ (alphabet size).

In the case of the Hamiltonian path problem, the harder instances actually have less literals and thus, on average, shorter clauses. This, too, is consistent with the nature of the problem in question: fewer edges (and thus shorter conditional clauses in the resultant formula) make for a graph that is harder to find a Hamiltonian path, while the most trivial satisfiable instance is actually one in which all possible edges exist.

Finally, in the comparison of ESCP instances, all analysed characteristics are virtually identical, further emphasising the lack of any clear pattern behind the relative difficulty of specific instances of SAT.

It is important to note that our analysis is not yet another attempt to back up the the findings previously described in Section 2, i.e., the existence of a phase transition and a 'constraint gap' at the boundary between regions of expected (un)satisfiability. Instead of considering the distribution of difficulty across some order parameter, we took into account benchmarks generated using the same settings, i.e., the same order parameter, in an attempt to pinpoint patterns related to the composition of the formulas themselves. However, it clearly appears that the differences are related to the specific characteristics of computational problems translated to SAT.

5 Conclusions

We have analysed Boolean formulas in CNF, representing translations of well-known NP-complete problems to SAT. The input files were grouped depending on their average processing time by SAT-solvers, and compared on several factors, including average clause length and percentage of Horn clauses, between the easiest and most difficult instances.

There do not appear to be easily noticeable global characteristics of CNF formulas representing harder instances of NP-complete problems. Depending on the specific problem and its translation to SAT, the formulas whose processing takes longer can, for instance, have longer clauses, or conversely, more clauses of similar average length to that in the 'easier' group. Similarly, the percentage of negative literals or Horn clauses

are also dependent on the NP-complete problem translated to SAT, and not some pattern prevalent across all comparatively easier or harder instances.

These observations seem in line with SAT being NP-complete, and as such, a difficult computational problem. Just as there is not a single SAT-solver always offering superior performance, no single factor contributes to a particular instance of SAT being comparatively easier or harder to verify than others of same size. This was most evident in the ESCP comparison: characteristics of both groups of instances were nearly identical, clearly showing that we cannot expect easy answers when it comes to hard computational problems. At least, not yet.

References

1. P. A. Abdulla, P. Bjesse, and N. Eén. Symbolic reachability analysis based on SAT-solvers. In *Proc. of the 6th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'00)*, volume 1785 of *LNCS*, pages 411–425. Springer-Verlag, 2000.
2. A. Armando and L. Compagna. An optimized intruder model for SAT-based model-checking of security protocols. In *Electronic Notes in Theoretical Computer Science*, volume 125, pages 91–108. Elsevier Science Publishers, March 2005.
3. G. Audemard and L. Simon. Glucose and syrup in the SAT race 2015. *SAT Race*, 2015.
4. A. Biere. Lingeling and friends entering the SAT challenge 2012. *Proceedings of SAT Challenge*, pages 33–34, 2012.
5. A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, pages 317–320, 1999.
6. P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, pages 331–337, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.
7. E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.
8. J. M. Crawford and L. D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81(1):31 – 57, 1996. Frontiers in Problem Solving: Phase Transitions and Complexity.
9. L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proc. of TACAS'08*, volume 4963 of *LNCS*, pages 337–340. Springer-Verlag, 2008.
10. M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. clasp : A conflict-driven answer set solver. In *LPNMR*, 2007.
11. I. P. Gent and T. Walsh. The hardest random SAT problems. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence*, pages 355–366, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.
12. Y. Hamadi, S. Jabbour, and L. Sais. ManySAT: a Parallel SAT Solver. *JSAT*, 6(4):245–262, 2009.
13. M. Kacprzak and W. Penczek. A SAT-based approach to unbounded model checking for alternating-time temporal epistemic logic. *Synthese*, 142:203–227, 2004.
14. R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
15. H. Kautz and B. Selman. Planning as satisfiability. In *ECAI 92: Proceedings of the 10th European conference on Artificial intelligence*, pages 359–363, 1992.

16. T. Larrabee and Y. Tsuji. Evidence for a satisfiability threshold for random 3CNF formulas. Technical report, 1993.
17. Y. Mahajan, Z. Fu, and S. Malik. Zchaff2004: An efficient sat solver. In *Int. Conf. on Theory and Applications of Satisfiability Testing*, pages 360–375. Springer, 2004.
18. A. Niewiadomski, W. Penczek, A. Półrola, M. Szreter, and A. Zbrzezny. Towards automatic composition of web services: SAT-based concretisation of abstract scenarios. *Fundam. Inform.*, 120(2):181–203, 2012.
19. A. Niewiadomski, W. Penczek, and T. Sidoruk. Comparing Efficiency of Modern SAT-solvers for Selected Problems in P, NP, PSPACE, and EXPTIME. In *Proceedings of the 26th International Workshop on Concurrency, Specification and Programming, Warsaw, Poland, September 25-27, 2017*, pages 1–12, 2017.
20. A. Niewiadomski, P. Switalski, W. Penczek, and T. Sidoruk. Applying Modern SAT-solvers to Solving Hard Problems. *Fundamenta Informaticae (submitted)*, 2018.
21. A. Niewiadomski, P. Switalski, W. Penczek, and T. Sidoruk. SMT-solvers in action: encoding and solving selected problems in NP and EXPTIME. *Scientific Annals of Computer Science (submitted)*, 2018.
22. N. Sorensson and N. Een. Minisat v1. 13 - a SAT solver with conflict-clause minimization. *SAT*, 2005(53):1–2, 2005.
23. J. S. Turner. Almost all k-colorable graphs are easy to color. *Journal of Algorithms*, 9(1):63 – 82, 1988.
24. B. Woźna, A. Zbrzezny, and W. Penczek. Checking reachability properties for Timed Automata via SAT. *Fundamenta Informaticae*, 55(2):223–241, 2003.
25. B. Woźna-Szczesniak. SAT-based bounded model checking for weighted deontic interpreted systems. *Fundam. Inform.*, 143(1-2):173–205, 2016.

Formal Semantics for Probabilistic Verification of Stochastic Regular Expressions

Sinem Getir[1], Esteban Pavese[1], and Lars Grunske[1]

Software Engineering, Humboldt University Berlin, Germany

Abstract. Modelling and verification of software systems is an effective phase of system development, as it can uncover failures in design early in the development process. There is an increasing need for languages and processes that allow for the specification of *uncertainty* that allow, for example, the modelling of the unknown behaviour of a user, or the stochastic failure rate of hardware components.
In this paper we introduce a formal semantics on Stochastic Regular Expressions (SREs) over probabilistic action logics for quantitative verification. We provide the recursive calculation of the language generated by an SRE, enhanced to reuse local results for global verification of system specifications. Furthermore, we demonstrate how to model systems with SREs and how to perform reachability analysis with Probabilistic Action-based Computational Tree Logic (PACTL*).

Keywords: stochastic regular expressions, probabilistic formalism, probabilistic model checking, probabilistic verification

1 Introduction

The analysis of systems with a probabilistic behaviour plays an important role in several applications, such as software engineering, speech recognition, digital communications and computational biology among others. On the other hand, regular expressions have spread through all of theoretical computer science and enjoy plentiful applications in the field of natural language processing, including parsing, deep language models, model inference and machine translation.

Several studies have been conducted in probabilistic version of regular expressions that is studied as probabilistic concurrent Kleene algebra in [12] and extended with additional Kleene theorems in the application of quantitative reasoning on database queries [3].

Kartzow and Weidner [7,17] define a Monadic Second-Order Logic (MSOL) and a constraint logic with temporal properties for data analysis for probabilistic regular expressions. Additionally, Weidner specified Probabilistic Regular Expressions for infinite strings with ω-properties in his thesis [16].

In this paper, we define a semantics of stochastic regular expressions in the context of probabilistic model checking, employing a probabilistic extension of Action-based Computation Tree Logic (ACTL*) to reason about temporal properties quantitatively. Probabilistic model checking [8] is a technique developed

in order to automatically perform such assessments, and has been successfully applied in recent years [10]. Probabilistic models used for model checking can be at different levels of abstraction, such as Markov Chains [2], Markov Decision Processes [2] and Stochastic Petri Nets [11] among others. In contrast to state-based representations, we introduce an approach and focus on stochastic regular expressions as an input model for probabilistic model checking applications.

2 Stochastic Regular Expressions (SRE)

In this section, we briefly recall SRE [14] and action logic [1].

2.1 Syntax of SRE

The syntax of a Stochastic Regular Expression (SRE) E over an alphabet Σ is defined recursively as follows:

$$E := \alpha \;\Big|\; \sum_i E_i[n_i] \;\Big|\; E_1 : E_2 \;\Big|\; E^{*f} \tag{1}$$

with $\alpha \in \Sigma \cup \{\varepsilon\}$, $n_i \in \mathbb{N}_0$, $f \in [0,1] \subset \mathbb{R}$ and every term E_i is a SRE, such that:

1. *Atomic Action* α: α is an atomic action that belongs to the alphabet Σ.
2. *Choice* $\sum_i E_i[n_i]$: One of the provided terms $E_i[n_i]$ is probabilistically chosen according to calculated probabilities from occurence values. n_i denotes the *occurrence value* or *choice rate* for each term, such that the i-th term is chosen with probability $\frac{n_i}{\sum_j n_j}$. *Occurrence value* or *choice rate* is defined as the number of cases that the node is chosen statistically.
3. *Concatenation* $E_1 : E_2$: The terms E_1 and E_2 are successively interpreted.
4. *Kleene Closure* E^{*f}: The term E is repeated for an indefinite number of times, subject to a binomial distribution. Each iteration occurs with a probability of f. The termination probability is $1 - f$.
5. *Plus Closure* E^{+f}: The *+Closure* is a syntactic sugar that is omitted here, but can be easily emulated with $E : E^{*f}$.

Without loss of generality, the empty string ε is not included in the alphabet. However we include the empty string ϵ as an atomic action. On the other hand, ϵ can be derived from an expression like $\alpha^{*0.0}, \alpha \in \Sigma$.

A derivation of a conventional regular expression E is the set of sentences, or strings over the alphabet, derivable from it. This defines the language L(E) of E. This notion of language derivation is similarly applicable to SRE, except that each string has a probability value associated with it, and hence the language itself is associated with a probability distribution of its members as explained in the following within its denotational semantics.

2.2 Denotational semantics of SREs

Intuitively SREs can be understood as an expression that defines a specific probability function over its language strings such that $[\![E]\!] : s \in \Sigma^* \rightarrow [0, 1]$. Such probabilistic language (p-language) is previously described for discrete events systems in [9]. In the following, we provide a trivial example on the p-language to explain the relationship between the SRE and the p-language.

The semantics of SREs are described by the p-language [9] in the style of denotational semantics [15]. The probability function for an SRE E is denoted by $[\![E]\!]$, and its application to a particular string s is denoted $[\![E]\!]s$, which represents an acceptance probability associated with string s in the language L(E). The probability function recursively calculates the occurrence probability of an arbitrary string $s \in \Sigma^*$ in a SRE model. By definition, if an arbitrary string s has a probability value greater than 0, it is accepted as a word of SRE. Meaning that if $s \in L(E)$ then $[\![E]\!]s > 0$.

Example 1. Let L be a probabilistic language [9] describing the Bernoulli process where each experiment has two outcomes a and b with probabilities p and $1 - p$ respectively. Then L is defined on the alphabet $\Sigma = \{a, b\}$ as $L(s) = p^{\#(a,s)}.(1 - p)^{\#(b,s)}$ where $\#(a, s)$ is representing the number of occurrences in word s.

In the following we formally define some additional notions which will be referred to throughout this paper.

Definition 1 (Words of a SRE). $Words(E) = \{w \mid w \in L(E)\}$

Definition 2 (Word length). *A **length** of a word $w = w_0w_1...w_n$ is the number of included characters and denoted as $|w| = n$ through the paper.*

Definition 3 (Probability of a word in the language). *The probability function $[\![E]\!]s$ for every possible SRE term E ($\alpha \mid \sum E_i(n_i) \mid E_1 : E_2 \mid E^{*f} \mid E^{+f}$) is calculated recursively, where $s = \alpha_1, ..\alpha_n \in \Sigma^*$:*

- ***Atomic actions:***

$$[\![\alpha]\!]s = 1, \textit{ if } \alpha = s$$
$$[\![\alpha]\!]s = 0, \textit{ if } \alpha \neq s \tag{2}$$

- ***Choice***

$$\Big[\!\Big[\sum E_i n_i \Big]\!\Big] s = \sum_k \Big(\frac{n_k}{\sum n_i} \Big) \cdot [\![E_k]\!]s \tag{3}$$

  Since every term might recognize s, the overall probability for a choice expression is the sum of all the term probabilities with respect to s.

- ***Concatenation**:*

$$[\![E_1 : E_2]\!]s = \sum_{i=0}^{n} ([\![E_1]\!]\alpha_1..\alpha_i \cdot [\![E_2]\!]\alpha_{i+1}..\alpha_n)$$

In the summation, s is decomposed into two (possibly empty) substrings, each of which may be consumed by a concatenated expression. Even though one term may recognize its substring argument, if the other term does not recognize its respective substring, then that term returns a probability of 0, and the overall probability for that instance of decomposition is 0.

– ***Kleene closure**:*

$$\begin{aligned} [\![E^{*f}]\!]\varepsilon &= 1 - f \\ [\![E^{*f}]\!]s &= \sum_{i=1}^{n} (f \cdot [\![E]\!]\alpha_1..\alpha_i \cdot [\![E^{*f}]\!]\alpha_{i+1}..\alpha_n) \\ &\quad + f \cdot [\![E]\!]s \cdot [\![E^{*f}]\!]\varepsilon \end{aligned} \tag{4}$$

The first formula accounts for empty strings, as the only way an iterated expression should recognize an empty string is by not iterating; in other words terminating without executing (The termination probability is therefore 1-f). The other formula recursively defines the general case. Here, one iteration of E will consume some portion of s, and the rest of s is consumed by further iterations. It is assumed that an iteration of a loop always consumes some non-empty string. Otherwise, the semantic model would have to account for Kleene closure iterating indefinitely on an argument, which is not an useful behaviour.

All SRE probability functions presented above are well formed probabilitity functions. Interested readers can find the details of probability functions and the proof of well formness in [14].

2.3 Action based Computation Tree Logic (ACTL*)

ACTL* is introduced in [4] where the comparison for state labelled and transition labelled systems is studied. The syntax of ACTL* is described recursively on action labelled transition systems as follows; where φ is a formula executed on the runs of the system:

$$\varphi := True \mid \neg\varphi \mid \varphi \wedge \varphi\prime \mid \exists\varphi \mid \varphi\, \mathcal{U}\, \varphi\prime \mid \mathcal{X}_a\varphi \mid \mathcal{X}\varphi \tag{5}$$

We biefly recall the definitions required for the ACTL* semantics.

Definition 4 (Labelled transition systems). *A labelled transition system is a tuple $(S, Act, \rightarrow)$ where:*

. *S is a finite set of states*
. *Act is a finite, non-empty set of actions*
. *$\rightarrow$ is the transition relation denoted in $\subseteq S \times (Act \cup \epsilon) \times S$ and any element of $\rightarrow$ is called a transition.*

Definition 5. *A sequence* $(s_0, \alpha_0, s_1)(s_1, \alpha_1, s_2).. \in \ \rightarrow \infty$ *is called a* ***path*** *from* s_0*. A* ***run*** $\rho = (s, \Phi)$ *is a pair from* $s \in S$*, where* Φ *is a path from* s*, first state of* ρ *is* s $(first(\rho) = s)$ *and* $path(\rho) = \Phi$*. If a* ***run*** θ *is a suffix for* ***run*** ρ*, then we denote as* $\theta \leq \rho$*.*

The semantics is given by satisfaction relations based on the definitions above:

$\rho \models True$ always
$\rho \models \neg\varphi$ iff $\rho \not\models \varphi$
$\rho \models \varphi \wedge \varphi'$ iff $\rho \models \varphi$ and $\rho \models \varphi'$
$\rho \models \exists\psi$ iff there exists a run $\theta \in run(first(\rho))$ such that $\theta \models \psi$
$\rho \models \psi \,\mathcal{U}\, \psi'$ iff there exists a θ with $\rho \leq \theta$ such that $\theta \models \psi\prime$ and for all $\rho \leq \eta \leq \theta : \eta \models \psi\prime$
$\rho \models \mathcal{X}\varphi$ iff there exists s, α, $s\prime$, θ such that $\rho = (s, (s, \alpha, s\prime))\theta$ and $\theta \models \varphi$
$\rho \models \mathcal{X}_a\varphi$ iff there exists s, $s\prime$, θ such that $\rho = (s, (s, a, s\prime))\theta$ and $\theta \models \varphi$ (6)

3 Semantics of Stochastic Regular Expression Trees with Probabilistic Action based Computation Tree Logic

Our goal is to reason about temporal properties on SRE models probabilistically. A very common logic Probabilistic computation tree logic (PCTL*) [6] and variants are defined on the state and path formulas. However Stochastic Regular Expressions do not have the explicit notation of a state. Therefore we prefer to extend the ACTL* logic semantically expressed on the **runs** of the system as provided in subsection 2.3.

The extended syntax of a Probabilistic ACTL* (PACTL*) is defined as follows where φ is a word and Φ is a SRE formula.

Definition 6 (Syntax of PACTL*).

$$\Phi = \neg\Phi \mid \Phi \wedge \Phi' \mid \mathcal{P}_P(\varphi) \tag{7}$$
$$\varphi = true \mid \mathcal{X}_a\varphi \mid \mathcal{X}\varphi \mid \varphi\,\mathcal{U}\,\varphi' \tag{8}$$

where $P \subseteq [0, 1]$

3.1 Semantics of PACTL*

Every SRE term E is defined as a node that specifies **operation type** (*choice, concatenation, kleene closure* or *action*)**, choice rate, kleene probability and subnode(s)**.

Formally E is an *action* or a tuple based on its type: $(\mathcal{N}, \mathcal{R})$, $(\mathcal{N})$, $(\mathcal{N}, k)$ if the opetaion types are choice, concatenation, kleene closure ($\mathcal{T} = + \mid : \mid *$ or action $a \in \Sigma$,) respectively. where $\mathcal{N} = \{E_1, E_2, ..., E_n\}$ is the finite set of subnodes in the operating order $(E_1 : E_2 \neq E_2 : E_1)$, $\mathcal{R} = \{r_1, r_2, ..r_n\}$ is the set of corresponding **choice rates** ($r \in \mathbb{N}$) and $k \in [0, 1]$ is a **kleene probability**.

We construct every SRE node in a bottom up fashion by parsing the given string by using operator presedence. Hence, we avoid the ambiguity of the parsed trees and remain them unique. (The order of operator predence is $*, :, +$). We also restrict one operation type per SRE node which allows unambiguity.

The semantics is defined a satisfaction relation for a word w and a SRE term E as follows:

$$E \models \neg\Phi \text{ iff } E \not\models \Phi \tag{9}$$
$$E \models \Phi \wedge \Phi' \text{ iff } E \models \Phi \wedge E \models \Phi' \tag{10}$$
$$E \models \mathcal{P}_P(\varphi) \text{ iff } Pr\{w \models \varphi \mid w \in Words(E)\} \in P \tag{11}$$
$$w \models true \text{ always} \tag{12}$$
$$w \models \mathcal{X}_a\varphi \text{ iff } w[0] = a \text{ and } w[1] \models \varphi \tag{13}$$
$$w \models \mathcal{X}\varphi \text{ iff } w[1] \models \varphi \tag{14}$$
$$w \models \varphi\,\mathcal{U}\,\varphi\prime \text{ iff for some } i \leq |w|,\ w[i] \models \varphi' \text{ and } w[j] \models \varphi,\ \forall j < i. \tag{15}$$

The set of words for an SRE can be calculated recursively on SRE node E:

$$Words(E) = \begin{cases} \{a\}, & \text{if } \mathcal{T} = a \\ \bigcup_{E_i \in \mathcal{N}}^{\infty} Words(E_i), & \text{if } \mathcal{T} = + \\ (Words(E_0) \cdot Words(E_1)) \ldots \cdot Words(E_n), \forall E_i \in \mathcal{N} & \text{if } \mathcal{T} =: \\ (Words(E_{\text{sub}}))^*, \text{ where } E_{\text{sub}} \in \mathcal{N} \text{ and } E = (E_{\text{sub}})^* & \text{if } \mathcal{T} = * \end{cases} \tag{16}$$

A system is specified as an SRE tree that includes a root node and finite set of nodes defined in the alphabet. An SRE tree is formally defined as; $T_E = (E_{\text{root}}, \mathcal{E}, \Sigma)$, where E_{root} is the root node, $\mathcal{E}$ is the finite set of all nodes and Σ is the alphabet. Hence, verifying the the root node E_{root} will result in verifying the system.

4 Example: System Specification with SRE Tree

We provide an example automata and a corresponding SRE system specification in the following paragraphs. Let us assume that we have a system that is composed of some web services aiming to achieve a message protocol. The subcomponents Service $1(S_1)$ and Service $2(S_2)$ are executing the login of sytem and message sending and logging out from the system respectively. The system can be defined as a stochastic regular expression tree as follows:

$T_E = (E_{\text{root}}, \mathcal{E}, \Sigma)$
$E_{\text{root}} = S$
$\mathcal{E} = \{S, S_1, S_2, E_1, E_2, E_3, E_4, E_5, E_6\}$
$\Sigma = \{start, login, authenticationFail, logout, sendMsg,$
$msgFail, terminate, success, retry\}$
$S = S_1 : S_2$
$S_1 = start : login$
$S_2 = authenticationFail[15] + logout[20] + E_1[65]$
$E_1 = sendMsg : E_2$
$E_2 = msgFail[5] + E_3[95]$
$E_3 = E_4 : E_5$
$E_4 = E_6^{*0.25}$
$E_5 = logout : terminate$
$E_6 = success : retry$

Let a PACTL* formula $\mathcal{P}_{[0.3,0.4]}\Big(true\,\mathcal{U}\,(\mathcal{X}_{\text{msgFail}})true\Big)$ for the analysis on T_E. The formula indicates the reachability analysis of the action "msgFail" on the root node $E_{\text{root}} = S$. The words reaching the "msgFail' from S are then recursively calculated:

$Words(S)_{[\text{reaching"msgFail"}]} \subset Words(S) = \{start.login.sendMsg.msgFail\ (0.0325),$
$start.login.sendMsg.succes.retry.msgFail\ (0.008125),$
$start.login.sendMsg.succes.retry.succes.retry.msgFail\ (0.00203125), ...\}$

The union is then

$$0.0325 \times \prod_{i=0}^{\infty}(0.25)^i = 0.0325 \in [0.3, 0.4] \tag{17}$$

$$S \models \mathcal{P}_{[0.3,0.4]}\Big(true\,\mathcal{U}\,(\mathcal{X}_{\text{msgFail}})true\Big) \tag{18}$$

Visually, we provide the corresponding probabilistic automata in Figure 1. The proof of equivalence between probabilistic Rabin automata [13] and the p-language, on which stochastic regular expression's semantics denoted, is provided in [5].

SREs are calculated in a bottom up way by remaining the probabilistic calculations of strings. Such techique enables to reach every calculation on each node locally. The idea is to calculate all information on every SRE term and compose the solutions based on the operations.

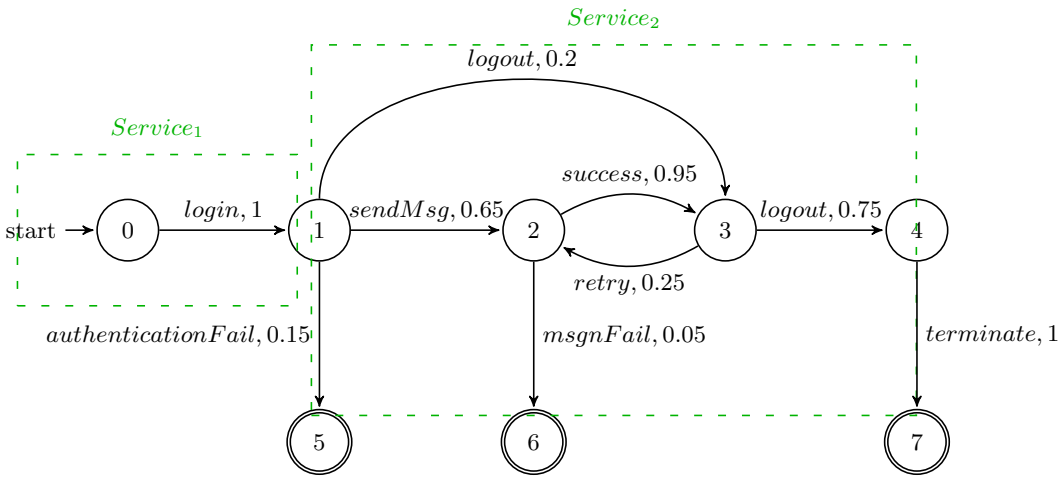


Fig. 1. Corresponding probabilistic automata to system T_E

5 Conclusion

We described a formal semantics for model checking of SREs that enjoys various applications in computer science. We studied the stochastic regular expressions with action based probabilistic logic in the model checking context and used stochastic regular expressions as an input model. Our initial attempt to reachability analysis with strings is also presented which is promising and convenient for parallel and incremental computation especially in the domain of component based systems or modular systems. The further investigation is to extend the reachibility analysis for the application of full PACTL* on SRE trees and evaluate our approach with set of models. Furthermore, we are planning to use SRE model checking for the incremental computation of local changes that can occur in the model.

References

1. Baier, C., Cloth, L., Haverkort, B.R., Kuntz, M., Siegle, M.: Model checking markov chains with actions and state labels. IEEE Trans. Software Eng. 33(4), 209–224 (2007)
2. Baier, C., Katoen, J.P.: Principles of Model Checking (Representation and Mind Series). The MIT Press (2008)
3. Bollig, B., Gastin, P., Monmege, B., Zeitoun, M.: A probabilistic kleene theorem. In: Chakraborty, S., Mukund, M. (eds.) Automated Technology for Verification and Analysis. pp. 400–415. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
4. De Nicola, R., Vaandrager, F.: Action versus state based logics for transition systems. In: Guessarian, I. (ed.) Semantics of Systems of Concurrent Processes. pp. 407–419. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)

5. Garg, V.K., Kumar, R., Marcus, S.I.: A probabilistic language formalism for stochastic discrete-event systems. IEEE Transactions on Automatic Control 44(2), 280–293 (Feb 1999)
6. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (Sep 1994)
7. Kartzow, A., Weidner, T.: Model checking constraint LTL over trees. CoRR abs/1504.06105 (2015)
8. Katoen, J.P.: The probabilistic model checking landscape. In: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 31–45. LICS '16, ACM, New York, NY, USA (2016)
9. Kumar, R., Garg, V.K.: Control of stochastic discrete event systems modeled by probabilistic languages. IEEE Trans. Automat. Contr. 46(4), 593–606 (2001)
10. Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic model checking in practice: Case studies with PRISM. ACM SIGMETRICS Performance Evaluation Review 32(4), 16–21 (2005)
11. Marsan, M.A.: Stochastic petri nets: An elementary introduction. In: Rozenberg, G. (ed.) Advances in Petri Nets 1989. pp. 1–29. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)
12. McIver, A., Rabehaja, T.M., Struth, G.: Probabilistic concurrent kleene algebra. In: Proceedings 11th International Workshop on Quantitative Aspects of Programming Languages and Systems, QAPL 2013, Rome, Italy, March 23-24, 2013. pp. 97–115 (2013)
13. Rabin, M.O.: Probabilistic automata. Information and Control 6(3), 230–245 (1963)
14. Ross, B.J.: Probabilistic pattern matching and the evolution of stochastic regular expressions. Applied Intelligence 13(3), 285–300 (Nov 2000)
15. Stoy, J.E.: Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory. MIT Press, Cambridge, MA, USA (1977)
16. Weidner, T.: Probabilistic Logic, Probabilistic Regular Expressions, and Constraint Temporal Logic. PhD dissertation, University Leipzig (2012)
17. Weidner, T.: Probabilistic Regular Expressions and MSO Logic on Finite Trees. In: Harsha, P., Ramalingam, G. (eds.) 35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015). Leibniz International Proceedings in Informatics (LIPIcs), vol. 45, pp. 503–516. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2015), http://drops.dagstuhl.de/opus/volltexte/2015/5630

More about left recursion in PEG

Roman R. Redziejowski

roman@redz.se

Abstract. Parsing Expression Grammar (PEG) is extended to handle left recursion, and under specified conditions becomes a correct parser for left-recursive grammars in Backus-Naur Form (BNF).

1 Introduction

The technique of parsing by recursive descent assigns to each grammatical construct a procedure that calls other procedures to process components of that construct. As grammars are usually defined in a recursive way, these calls are recursive. This method encounters two problems:

(1) Procedures corresponding to certain type of construct must chose which procedure to call next.

(2) If the grammar contains left recursion, a procedure may call itself indefinitely.

Problem (1) has been traditionally solved by looking at the next input symbol(s), which works if the grammar satisfies a condition known as LL(n). Another way is to try the alternatives one by one, backtracking in the input, until one succeeds (or all fail).

Making full search can require exponential time, so a possible option is limited backtracking: never return after a partial success. This method has been used in actual parsers [4,10] and is described in literature [1–3,7]. It has been eventually formalized by Ford [5] under the name of Parsing Expression Grammar (PEG).

Problem (2) is serious because of a strong tendency to present grammars in left-recursive form. Converting the grammar to right-recursive form is possible, but is tedious, error-prone, and obscures the spirit of the grammar.

The problem has been in the recent years solved by what can be called "recursive ascent". Each recursion must end up in a part of syntax tree that does not involve further recursion. It has been referred to as the "seed". After identifying the seed, one reconstructs the syntax tree upwards, in the process of "growing the seed". Extensions to PEG using this method have been described in [6,8,12,15–17].

The paper tries to find out under which conditions this process will work correctly. The idea of "working correctly" needs an explanation. One of the most common ways to define the syntax of a formal language is the Backus-Naur Form (BNF) or its extended version EBNF. We treat here PEG as a parser for BNF that implements recursive descent with limited backtracking. Because of limited backtracking, PEG may miss some strings that belong to the language defined by BNF. The author has previously tried to answer the question under which

conditions PEG will accept exactly the language defined by BNF (see [13, 14]). This paper tries to answer the same question for PEG equipped with recursive ascent technique to handle left recursion.

We look here at a process inspired by Hill [6]. Section 2 introduces a subset of BNF grammar with natural semantics that is a slight modification of that due to Medeiros [9, 11]. In Section 3 we develop some concepts needed to discuss left recursion. In Section 4 we describe the parsing process, check that it terminates, and state the conditions under which it reproduces the BNF syntax tree. The last section contains some comments. Proofs of the Propositions are found in Appendix.

2 The BNF grammar

We consider an extremely simplified form of BNF grammar over alphabet Σ. It is a set of *rules* of the form $A = e$ where A belongs to a set N of symbols distinct from the letters of Σ and e is an *expression*. Each expression is one of these:

$$\begin{array}{ll} \varepsilon \text{ (”empty”)}, & e_1 e_2 \text{ (”sequence”)}, \\ a \in \Sigma \text{ (”letter”)}, & e_1 | e_2 \text{ (”choice”)}. \\ A \in N \text{ (”nonterminal”)}. & \end{array}$$

where each of e_1, e_2 is an expression and ε denotes empty word. The set of all expressions is in the following denoted by $\mathbb{E}$. There is exactly one rule $A = e$ for each $A \in N$. The expression e appearing in this rule is denoted by $e(A)$.

Each expression $e \in \mathbb{E}$ has its language $\mathcal{L}(e) \subseteq \Sigma^*$ defined formally by natural semantics shown in Figure 1. String x belongs to $\mathcal{L}(e)$ if and only if $[e]x \overset{\text{BNF}}{\to} x$ can be proved using the inference rules from Figure 1. Note that if a proof of $[e]x \overset{\text{BNF}}{\to} x$ exists, one can prove $[e]xy \overset{\text{BNF}}{\to} x$ for every $y \in \Sigma^*$

$$\frac{}{[\varepsilon]w \overset{\text{BNF}}{\to} \varepsilon}\ (\textit{empty}) \qquad \frac{}{[a]aw \overset{\text{BNF}}{\to} a}\ (\textit{letter}) \qquad \frac{[e(A)]w \overset{\text{BNF}}{\to} x}{[A]w \overset{\text{BNF}}{\to} x}\ (\textit{rule})$$

$$\frac{[e_1]xw \overset{\text{BNF}}{\to} x \quad [e_2]w \overset{\text{BNF}}{\to} y}{[e_1 e_2]xw \overset{\text{BNF}}{\to} xy}\ (\textit{seq})$$

$$\frac{[e_1]w \overset{\text{BNF}}{\to} x}{[e_1|e_2]w \overset{\text{BNF}}{\to} x}\ (\textit{choice1}) \qquad \frac{[e_2]w \overset{\text{BNF}}{\to} x}{[e_1|e_2]w \overset{\text{BNF}}{\to} x}\ (\textit{choice2})$$

Fig. 1. Formal semantics of BNF

One can read $[e]w \overset{\text{BNF}}{\to} x$ as saying that expression e matches prefix x of string w. Thus, $x \in \mathcal{L}(e)$ means that e matches the string x. We say that expression e is *nullable* to mean that $\varepsilon \in \mathcal{L}(e)$.

Figure 2 is an example of formal proof using the rules from Figure 1. It verifies that the string *baac* belongs to $\mathcal{L}(S)$ as defined by this grammar:

```
S = Ac    A = Aa|B    B = b
```

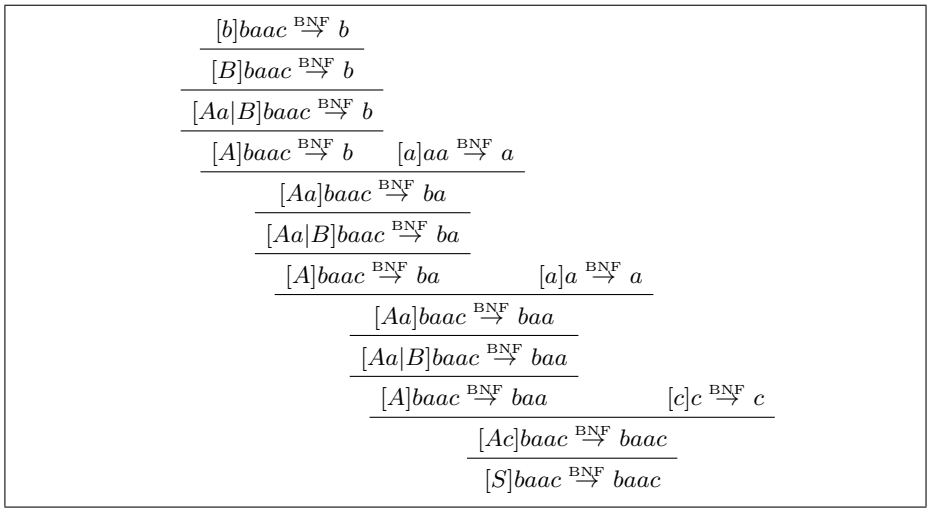


Fig. 2. Example of BNF proof

The proof can be represented in the inverted form shown in Figure 3. This seems more intuitive when speaking about "recursive descent". The diagram on the right is simplified to show only the expressions. It is the syntax tree of *baac*.

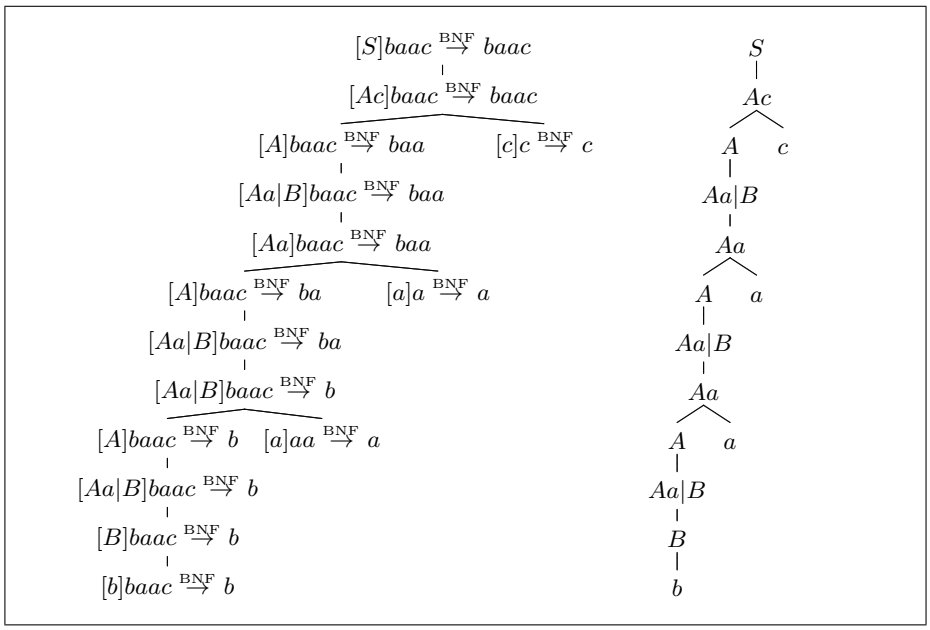


Fig. 3. BNF tree and syntax tree

3 Left recursion

3.1 Recursion classes

For $e \in \mathbb{E}$ define $\texttt{first}(e)$ as follows:

$$\texttt{first}(\varepsilon) = \texttt{first}(a) = \varnothing, \qquad \texttt{first}(A) = \{e(A)\},$$
$$\texttt{first}(e_1|e_2) = \{e_1, e_2\}, \qquad \texttt{first}(e_1e_2) = \begin{cases} \{e_1\} \text{ if } \varepsilon \notin \mathcal{L}(e_1), \\ \{e_1, e_2\} \text{ if } \varepsilon \in \mathcal{L}(e_1). \end{cases}$$

Define further $\texttt{First}$ to be the transitive closure of relation $\texttt{first}$. In the following, we write $e \xrightarrow{\text{first}} e'$ to mean that $e' \in \texttt{first}(e)$, and $e \xrightarrow{\text{First}} e'$ to mean that $e' \in \texttt{First}(e)$.

An expression e is *left-recursive* if $e \xrightarrow{\text{First}} e$. Let $\mathbb{R} \subseteq \mathbb{E}$ be the set of all left-recursive expressions. Define relation $\texttt{Rec} \subseteq \mathbb{R} \times \mathbb{R}$ so that $(e_1, e_2) \in \texttt{Rec}$ means $e_1 \xrightarrow{\text{First}} e_2 \xrightarrow{\text{First}} e_1$. It is an equivalence relation that defines a partition of $\mathbb{R}$ into equivalence classes; we refer to them as *recursion classes*. The recursion class containing expression e is denoted by $\mathbb{C}(e)$.

Let e be an expression belonging to recursion class $\mathbb{C}$. If $e = A \in N$ or $e = e_1e_2$ with non-nullable e_1, the expression $e' \in \texttt{first}(e)$ must also belong to $\mathbb{C}$. This is so because e' is the only element in $\texttt{first}(e)$, and we must have $e \xrightarrow{\text{first}} e' \xrightarrow{\text{First}} e$ to achieve $e \xrightarrow{\text{First}} e$. In $e = e_1|e_2$ or $e = e_1e_2$ with nullable e_1, one of expressions e_1 or e_2 may be outside $\mathbb{C}$. It is a *seed* of $\mathbb{C}$, and e is an *exit* of $\mathbb{C}$. For $e = e_1e_2$ in $\mathbb{C}$, e_2 is a *leaf* of $\mathbb{C}$. The set of all seeds of $\mathbb{C}$ is denoted by $\texttt{Seed}(\mathbb{C})$ and the set of all its leafs by $\texttt{Leaf}(\mathbb{C})$.

As an example, the grammar used in Figure 2 has $\mathbb{R} = \{A, Aa|b, Aa\}$. All these expressions belong to the same recursion class with exit $Aa|b$, seed b and leaf a.

To simplify the discussion, we asume in the following that all exits have the form $e_1|e_2$, that is, $e_1 \in \mathbb{R}$ in e_1e_2 is not nullable. As the ordering of BNF choice expression does not influence its language, we assume that e_2 in $e_1|e_2$ is always the seed.

3.2 Recursion sequence

Consider a node in the syntax tree and the leftmost branch emanating from it. It is a chain of nodes connected by $\xrightarrow{\text{first}}$. If the branch contains any left-recursive expressions, expressions belonging to the same recursion class $\mathbb{C}$ must form an uninterrupted sequence. Indeed, if e_1 and e_2 appearing in the branch belong to $\mathbb{C}$, we have $e_1 \xrightarrow{\text{First}} e \xrightarrow{\text{First}} e_2 \xrightarrow{\text{First}} e_1$ for any e in between. Such sequence is a *recursion sequence* of class $\mathbb{C}$. The same argument shows that the branch can contain at most one recursion sequence of a given class, and that sequences of different classes cannot be nested.

The last expression in the sequence must be an exit of $\mathbb{C}$. One can easily see that each recursion class must have at least one exit. We assume in the following that this is the case.
Let $e \xrightarrow{\text{first}} e'$ be two consecutive expressions in a recursion sequence. Let $[e]w \overset{\text{BNF}}{\to} x$ and $[e']w \overset{\text{BNF}}{\to} x'$. Expression e can be one of these:

- $A = e'$. Then $x = x'$ according to (*rule*).
- $e'|e_2$. Then $x = x'$ according to (*choice1*).
- $e_1|e'$. Then $x = x'$ according to (*choice2*).
- $e'e_2$. Then $x = x'y$ where $y \in \mathcal{L}(e_2)$ according to (*seq*).

Define $\texttt{adv}(e, e') = \{\varepsilon\}$ in each of the first three cases and $\texttt{adv}(e, e') = \mathcal{L}(e_2)$ in the last. For a recursion sequence $s = e_{[n]}, e_{[n-1]}, \ldots, e_{[2]}, e_{[1]}$ define

$$\texttt{adv}(s) = \texttt{adv}(e_{[n]}, e_{[n-1]}) \ldots \texttt{adv}(e_{[2]}, e_{[1]}).$$

For $e \in \mathbb{R}$ define $\texttt{Adv}(e)$ to be the union of $\texttt{adv}(s)$ for all recursion sequences s starting with e, ending with e, and not containing e. The following is easy to see:

Lemma 1. *If $[e]w \overset{BNF}{\to} x$ and $[e]w \overset{BNF}{\to} x'$ are two consecutive results for the same e in a recursion sequence, we have $x \in x'\texttt{Adv}(e)$.*

4 Parsing

Given a BNF grammar, we define for each $e \in \mathbb{E}$ a parsing procedure named $[e]$. The procedure may return "success" after possibly "consuming" some input, or "failure" without consuming anything. In the following, the result of calling $[e]$ for input w is denoted by $[e]w$; it is either `fail` or the consumed string x (possibly ε).

The action of parsing procedure $[e]$ for $e \notin \mathbb{R}$ is the same as in PEG and is shown in Figure 4.

The procedure $[e]$ for $e \in \mathbb{R}$ is a "grower" for the recursion class $\mathbb{C}(e)$ and input string w. The grower has a "plant" $[e', w]$ for each expression $e' \in \mathbb{C}(e)$. The plant is a procedure with memory. The procedure emulates the action of parsing procedure $[e']$, but may use results from other plants instead of calls to parsing procedures. It computes a result as shown in Figure 5. The memory holds the result of procedure applied to w. It is denoted by $\langle e', w\rangle$. The grower initializes all its plants with $\langle e', w\rangle = \texttt{fail}$, and then repeatedly calls all plants in some order. If the result is better than already held by the plant, it replaces the latter. (A string is better than `fail`, and longer string is better than shorter one.) The grower stops when it cannot improve any result. The result in $[e, w]$ is then the result of parsing procedure $[e]$.

In order to create a formal record of parsing process, we represent actions of parsing procedures and plants by inference rules shown in Figure 6. A rule with conclusion $[e]w = X$ represents a call to $[e]$ returning X. One with conclusion $\langle e, w\rangle = x$ represents setting new result x in $\langle e, w\rangle$. A premise $[e']w = X$

represents call a to sub-procedure $[e']$ returning X; a premise $\langle e', w\rangle = X$ represents result obtained from $[e', w]$. With these conventions, parsing process is represented as a formal proof.

An example of such proof is shown in Figure 7. It represents parsing process for the string and grammar from example in Figure 3. Note that part of that tree was constructed by going down and part by going up.

$[\varepsilon]$	Indicate success without consuming any input.
$[a]$	If the text ahead starts with a, consume a and return success. Otherwise return failure.
$[A = e_1]$	Call $[e_1]$ and return result.
$[e_1\, e_2]$	Call $[e_1]$. If it succeeded, call $[e_2]$ and return success if $[e_2]$ succeeded. If $[e_1]$ or $[e_2]$ failed, backtrack: reset the input as it was before the invocation of $[e_1]$ and return failure.
$[e_1 \mid e_2]$	Call $[e_1]$. Return success if it succeeded. Otherwise call $[e_2]$ and return success if $[e_2]$ succeeded or failure if it failed.

Fig. 4. Actions of parsing procedures

$[A, w]$	Return $\langle e(A), w\rangle$.
$[(e_1\, e_2), w]$	If $\langle e_1, w\rangle = \texttt{fail}$, return `fail`. If $\langle e_1, w\rangle = x$, call $[e_2]$ on z where $w = xz$ and restore input to w. If $[e_2]z = \texttt{fail}$, return `fail`. Otherwise return $x\,[e_2]z$.
$[(e_1 \mid e_2), w]$	If $\langle e_1, w\rangle \neq \texttt{fail}$, return $\langle e_1, w\rangle$. If $\langle e_1, w\rangle = \texttt{fail}$ and there exists plant $[e_2, w]$, return $\langle e_2, w\rangle$. Otherwise call $[e_2]$, restore input to w, and return $[e_2]w$.

Fig. 5. Actions of plants

We say that "parsing procedure $[e]$ handles string w" to mean that the procedure applied to w terminates and returns either $x \in \Sigma^*$ or `fail`.

Proposition 1. *Each parsing procedure* $[e]$ *for* $e \in \mathbb{E}$ *handles all* $w \in \Sigma^*$.

Proof is found in the Appendix.

Proposition 2. *For each result* $[e]w = x$ *or* $\langle e, w\rangle = x$ *in a parse tree there exists a proof of* $[e]w \overset{BNF}{\rightarrow} x$.

Proof is by induction on height of the subtree.

The conditions under which each BNF tree has a corresponding parse tree depend on the context in which certain expression may be used. We assume that the grammar has a start expression S and use as context the BNF proof for $[S]u \overset{\text{BNF}}{\to} u$ for $u \in \Sigma^*$.

For $e \in \mathbb{E}$, we define $\texttt{Tail}(e)$ as the set of all strings that may follow a string matched by e in the proof of $[S]u \overset{\text{BNF}}{\to} u$. More precisely, it is the set of strings z in all results $[e]xz \overset{\text{BNF}}{\to} x$ that appear in that proof. A possible method for estimating $\texttt{Tail}(e)$ can be found in [14]. For $e \in \mathbb{R}$, $\texttt{Tail}_R(e)$ excludes occurrences of e in a recursion path except the first one.

The ordering of choice expression is essential for rules (*choice2.p*), (*choice2.g*), and(*choice3.g*). But, the ordering does not affect the language defined by BNF rules. Therefore, we can always rearrange the choice as is best for the parsing.

Proposition 3. *If the grammar satisfies the following conditions (1)-(3) then for each proof of* $[S]u \overset{BNF}{\to} u$ *there exists parse tree with root* $[S]u = u$*. Moreover, for each subproof* $[e]w \overset{BNF}{\to} x$ *of that proof exists parse tree with root* $[e]w = x$ *or* $\langle e, w\rangle = x$*.*

$$\textit{For each}\ \ e = e_1|e_2 \in \mathbb{R},\ \ e_2 \notin \mathbb{C}(e); \tag{1}$$

$$\textit{For each}\ \ e = e_1|e_2,\ \ \mathcal{L}(e_1)\Sigma^* \cap \mathcal{L}(e_2)\mathit{Tail}(e) = \varnothing; \tag{2}$$

$$\textit{For each}\ \ e \in \mathbb{R},\ \ \mathit{Adv}(e)\Sigma^* \cap \mathit{Tail}_R(e) = \varnothing. \tag{3}$$

Proof is found in the Appendix.

5 Comments

We presented sufficient conditions under which the extended PEG is a correct parser for BNF grammars. Because we treat PEG as parser for BNF, it does not include syntactic predicates of classical PEG.

We chose here the scheme for handling left recursion inspired by [6] because it seems easy to analyze. The scheme where the grower scans all plants even if nothing changes is also easy to analyze; in a practical implementation the plant would be called only if its argument(s) change.

Checking (2) in the presence of left recursion is not simple. Without left recursion, one can use approximation by prefixes, with LL(1) as the extreme case. The languages defined by left recursion often have identical prefixes and differ only at the far end.

The chosen scheme required a rather severe restriction (1) to the grammar. It seems possible to replace it by a requirement that languages of different seeds of the same recursion class are disjoint, as well as languages of expressions in $\texttt{first}^{-1}(e)$ for $e \in \mathbb{R}$. This is the subject of further research, as well as attempts to analyze other schemes.

$$\frac{}{[\varepsilon]w = \varepsilon}\;(\textit{empty.p}) \qquad \frac{[e(A)]w = X}{[A]w = X}\;(\textit{rule.p}) \qquad \frac{\langle e, w\rangle = X}{[e]w = X}\;(\textit{grow.p})$$

$$\frac{}{[a]aw = a}\;(\textit{letter1.p}) \qquad \frac{b \neq a}{[b]aw = \texttt{fail}}\;(\textit{letter2.p}) \qquad \frac{}{[a]\varepsilon = \texttt{fail}}\;(\textit{letter3.p})$$

$$\frac{[e_1]xz = x \quad [e_2]z = y}{[e_1e_2]xz = xy}\;(\textit{seq1.p}) \qquad \frac{[e_1]w = \texttt{fail}}{[e_1e_2]w = \texttt{fail}}\;(\textit{seq2.p})$$

$$\frac{[e_1]xz = x \quad [e_2]z = \texttt{fail}}{[e_1e_2]xz = \texttt{fail}}\;(\textit{seq3.p})$$

$$\frac{[e_1]w = x}{[e_1 \,|\, e_2]w = x}\;(\textit{choice1.p}) \qquad \frac{[e_1]w = \texttt{fail} \quad [e_2]w = X}{[e_1 \,|\, e_2]w = X}\;(\textit{choice2.p})$$

$$\frac{\langle e(A), w\rangle = X}{\langle A, w\rangle = X}\;(\textit{rule.g}) \qquad \frac{\langle e_1, xz\rangle = x \quad [e_2]z = y}{\langle (e_1e_2), xz\rangle = xy}\;(\textit{seq1.g})$$

$$\frac{\langle e_1, xz\rangle = x \quad [e_2]z = \texttt{fail}}{\langle (e_1e_2), xz\rangle = \texttt{fail}}\;(\textit{seq2.g}) \qquad \frac{\langle e_1, w\rangle = \texttt{fail}}{\langle (e_1e_2), w\rangle = \texttt{fail}}\;(\textit{seq3.g})$$

$$\frac{\langle e_1, w\rangle = x}{\langle (e_1 \,|\, e_2), w\rangle = x}\;(\textit{choice1.g}) \qquad \frac{\langle e_1, w\rangle = \texttt{fail} \quad \langle e_2, w\rangle = X}{\langle (e_1 \,|\, e_2), w\rangle = X}\;(\textit{choice2.g})$$

$$\frac{\langle e_1, w\rangle = \texttt{fail} \quad [e_2]w = X}{\langle (e_1 \,|\, e_2), w\rangle = X}\;(\textit{choice3.g}) \qquad \frac{e \in \mathbb{R}}{\langle e_1, w\rangle = \texttt{fail}}\;(\textit{init.g})$$

where X denotes x or $\texttt{fail}$.

Fig. 6. Formal semantics of parser

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\langle Aa, baac\rangle = \texttt{fail} \quad \dfrac{[b]baac = b}{[B]baac = b}}{\langle Aa|B, baac\rangle = b}}{\langle A, baac\rangle = b} \quad [a]aac = a}{\langle Aa, baac\rangle = ba}}{\langle Aa|B, baac\rangle = ba}}{\langle A, baac\rangle = ba} \quad [a]ac = a}{\langle Aa, baac\rangle = baa}}{\langle Aa|B, baac\rangle = baa}}{\langle A, baac\rangle = baa}}{[A]baac = baa} \quad [c]c = c}{[Ac]baac = baac}}{[S]baac = baac}$$

Fig. 7. Example of parse tree

A Appendix

A.1 Proof of Proposition 1

The proof is by induction on the length of w with Lemma 3 as induction base and Lemma 2 as induction step. □

Lemma 2. *If each parsing procedure handles all words of length n or less, each parsing procedure handles all words of length $n+1$.*

Proof. Let the *rank* of expression e, denoted $\rho(e)$, be defined as follows:

- $\rho(\varepsilon) = \rho(a) = 0$, and otherwise:
- For $e \notin \mathbb{R}$, highest $\rho(e')$ for $e' \in \texttt{First}(e)$ plus 1;
- For $e \in \mathbb{R}$, highest $\rho(e')$ for $e' \in \texttt{Seed}(\mathbb{C}(e)) \cup \texttt{Leaf}(\mathbb{C}(e))$ plus 1.

One can easily see that this definition is not circular.

Assume that each procedure handles all words of length n or less, and consider a word w of length $n+1$. We use induction on rank of e to show that each $[e]$ handles w.

(Induction base:) Obviously, each procedure of rank 0 handles w.

(Induction step:) Assume that each procedure of rank m or less handles all words of length $n+1$ or less. Take any procedure $[e]$ of rank $m+1$.
If $e \notin \mathbb{R}$, e can be one of these:

- $A \in N$. We have $\rho(e(A)) = m$; thus $e(A)$ handles w, and so does A.
- $e_1 e_2$ with nullable e_1. We have $\rho(e_1) \leq m$ and $\rho(e_2) \leq m$. Each of them handles words of length $n+1$ or less, so $e_1 e_2$ handles w.
- $e_1 e_2$ with non-nullable e_1. We have $\rho(e_1) = m$, so e_1 handles w. If it fails, so does $e_1 e_2$. Otherwise it consumes $x \neq \varepsilon$ and e_2 is applied to the rest w' of w, with length n or less. Thus, e_2 handles w', so $e_1 e_2$ handles w.
- $e_1 | e_2$. We have $\rho(e_1) \leq m$ and $\rho(e_2) \leq m$. Each of them handles words of length $n+1$ or less, so $e_1 | e_2$ handles w.

If $e \in \mathbb{R}$, the result of $[e]$ is obtained by grower for the recursion class $\mathbb{C}(e)$ and input w. The grower stops when it cannot improve any result. Each improvement means consuming more of w. Since w is finite, the grower must eventually stop. Thus, $[e]$ handles w. □

Lemma 3. *Each parsing procedure handles word of length 0.*

Proof. The proof is essentially the same as that of Lemma 2, with w replaced by ε, and simplified case of $e_1 e_2$ with non-nullable e_1. □

A.2 Proof of Proposition 3

Suppose we are given a BNF proof of $[S]u \overset{\text{BNF}}{\to} u$. We are going to show that each partial proof of that proof (and the final proof) has the corresponding parse tree. In the following, we say "subtree $[e]w \overset{\text{BNF}}{\to} x$" to mean the partial proof with result $[e]w \overset{\text{BNF}}{\to} x$. Define the *level* of subtree $[e]w \overset{\text{BNF}}{\to} x$ as follows:

– For $e = \varepsilon$ and $e = a$ the level is 1.
– For $e \notin \mathbb{R}$, other than above, the level is 1 plus the highest level of subtrees for components of e.
– Each result with $e \in \mathbb{R}$ belongs to some recursion sequence. All subtrees in that sequence have the same level, equal to 1 plus the highest level of subtrees for the seed and leafs of that recursion sequence.

The proof is by induction on the level.

(Induction base:) The parse trees for subtrees on level 1 are $[\varepsilon]w = \varepsilon$ respectively $[a]az = a$. They represent calls to parsing procedures $[\varepsilon]$ and $[a]$.

(Induction step:) Assume that there exists parse tree for each subtree on level n or less. We show that there exists parse tree for each subtree on level $n+1$.
For a subtree $[e]w \overset{\text{BNF}}{\to} x$ on level $n+1$ where $e \notin \mathbb{R}$ we construct parse tree from parse trees of subtrees. This is done in Lemma 4 and represents calls from $[e]$ to its subprocedures.
The root of each subtree $[e]w \overset{\text{BNF}}{\to} x$ on level $n+1$ where $e \in \mathbb{R}$ belongs to some recursion sequence (perhaps degenerated to length 1). We construct parse trees for all results in the sequence from parse trees for the leafs and the seed. This is done in Lemma 5 and represents work done by the grower. □

Lemma 4. *Assume there exists parse tree for each subtree of* $[e]w \overset{BNF}{\to} x$. *There exists parse tree for* $[e]w \overset{BNF}{\to} x$.

Proof. The parse tree for $[e]w \overset{\text{BNF}}{\to} x$ is constructed in the way that depends on e:

– $A = e'$. $[A]w \overset{\text{BNF}}{\to} x$ is derived from $[e']w \overset{\text{BNF}}{\to} x$ according to (*rule*).
As assumed, there exists parse tree $[e']w = x$ for $[e']w \overset{\text{BNF}}{\to} x$. The parse tree $[A]w = x$ is constructed from it using (*rule.p*).
– e_1e_2. $[e_1e_2]xz \overset{\text{BNF}}{\to} xy$ is derived from $[e_1]xz \overset{\text{BNF}}{\to} x$ and $[e_2]z \overset{\text{BNF}}{\to} y$ according to (*seq*).
As assumed, there exist parse trees $[e_1]xz = x$ and $[e_2]z = y$. for $[e_1)]xz \overset{\text{BNF}}{\to} x$ and $[e_2]z \overset{\text{BNF}}{\to} y$. The parse tree $[e_1e_2]xz = xy$ is built from them using (*seq1.p*).
– $e_1|e_2$ with $[e_1|e_2]w \overset{\text{BNF}}{\to} x$ derived from $[e_1]w \overset{\text{BNF}}{\to} x$ according to (*choice1*).
As assumed, there exists parse tree $[e_1]w = x$ for $[e_1]w \overset{\text{BNF}}{\to} x$. The parse tree $[e_1|e_2]w = x$ is constructed from it using (*choice1.p*).
– $e_1|e_2$ with $[e_1|e_2]w \overset{\text{BNF}}{\to} x$ derived from $[e_2]w \overset{\text{BNF}}{\to} x$ according to (*choice2*).
As assumed, there exists parse tree $[e_2]w = x$ for $[e_2]w \overset{\text{BNF}}{\to} x$. $[e_2]w \overset{\text{BNF}}{\to} x$ means $w \in \mathcal{L}(e_2)\texttt{Tail}(e)$. As specified in Figure 4, parser calls the procedure $[e_1]$ on z. According to Proposition 1, the call returns either `fail` or prefix y of w. By Proposition 2, this latter would mean $w \in \mathcal{L}(e_1)\Sigma^*$, which contradicts (2). Therefore must be $[e_1]y = \texttt{fail}$. The parse tree $[e_1|e_2]w = x$ is constructed from $[e_1]w = \texttt{fail}$ and $[e_2]w = x$ using (*choice2.p*).

□

Lemma 5. *If there exist parse tree for each seed and each leaf of recursion sequence $e_{[k]}, \ldots, e_{[2]}, e_{[1]}$, there exists parse tree for each result in the sequence and in particular for $e = e_{[k]}$.*

Proof. Suppose the grower is called to handle e for input w. We start by showing that after the n-th round of the grower, $\langle e_{[n]} \rangle$ contains the root of parse tree for the subtree $[e_{[n]}]w \overset{\text{BNF}}{\rightarrow} x_{[n]}$.

As the grower checks all plants on each round, it will call $\langle e_{[n]} \rangle$ on round n. The proof is by induction on n.

(Induction base:) Expression $e_{[1]}$ is an exit $e_1|e_2$ of the sequence, with e_2 as seed. The result $[e_{[1]}]w \overset{\text{BNF}}{\rightarrow} x$ is derived from $[e_2]w \overset{\text{BNF}}{\rightarrow} x$ using (*choice2*).
When the grower calls $[e_{[1]}, z]$ in its first round, $[e_1, z]$ contains `fail`. As e_2 is the seed of the sequence, there exists parse tree with root $[e_2]w = x$. The parse tree for $\langle e_{[1]}, w \rangle = x$ is constructed from it using (*choice3.g*).

(Induction step:) Consider the round $n+1$ and assume that $[e_{[n]}, w]$ contains the root $\langle e_{[n]}, w \rangle = x$ of parse tree for subtree $[e_{[n]}]w \overset{\text{BNF}}{\rightarrow} x$. What happens when the grower calls $[e_{[n+1]}, w]$ depends on expression $e_{[n+1]}$. It can be one of these:

- $A = e'$ with $[e_{[n+1]}]w \overset{\text{BNF}}{\rightarrow} x$ derived from $[e']w \overset{\text{BNF}}{\rightarrow} x$ according to (*rule*).
  The e' here is $e_{[n]}$ with parse tree $\langle e', w \rangle = x$. The parse tree for $\langle e_{[n+1]}, w \rangle$ is constructed from it using (*rule.g*).
- $e_1 e_2$ with $[e_{[n+1]}]xz \overset{\text{BNF}}{\rightarrow} xy$ derived from $[e_1]xz \overset{\text{BNF}}{\rightarrow} x$ and $[e_2]z \overset{\text{BNF}}{\rightarrow} y$ according to (*seq*).
  The e_1 here is $e_{[n]}$ with parse tree $\langle e_1, xz \rangle = x$. As e_2 is a leaf of the sequence, there exists parse tree with root $[e_2]z = y$. The parse tree for $[e_{[n+1]}, w]$ is constructed using (*seq1.g*).
- $e_1|e_2$. By (1), $e_2 \notin \mathbb{C}$. The BNF result $[e_{[n+1]}]w \overset{\text{BNF}}{\rightarrow} x$ is derived from or $[e_2]w \overset{\text{BNF}}{\rightarrow} x$ according to (*choice2*).
  As $e_{[n]}$ is in $\mathbb{C}$, it must be e_1 with parse tree $\langle e_1, w \rangle = x$. The parse tree for $\langle e_{[n+1]}, w \rangle$ is constructed from it using (*choice1.g*).

Note that from (3) follows $\varepsilon \notin \texttt{Adv}(e)$. Thus, according to Lemma 1, a new result is stored in $\langle e_{[n]}, w \rangle$ in every turn.

Suppose the grower does not stop after $e_{[k]}$ because it finds a better result for plant $[e_{[k]}, w]$. If this better result is x', we have by Lemma 1 $x' = \langle e_{[k]}, w \rangle \texttt{Adv}(e_{[k]})$ which means $\texttt{Adv}(e_{[k]})$ is a prefix of $\texttt{Tail}_R(e)$, and contradicts (3). Thus, the grower stops at $e_{[k]}$.

The parse tree $[e]w = x$ is constructed from $\langle e_{[k]}, w \rangle = x$ using (*grow.p*). □

References

1. Aho, A.V., Sethi, R., Ullman, J.D.: Compilers, Principles,Techniques, and Tools. Addison-Wesley (1987)
2. Birman, A.: The TMG Recognition Schema. Ph.D. thesis, Princeton University (February 1970)
3. Birman, A., Ullman, J.D.: Parsing algorithms with backtrack. Information and Control 23, 1–34 (1973)
4. Brooker, P., Morris, D.: A general translation program for phrase structure languages. J. ACM 9(1), 1–10 (1962)
5. Ford, B.: Parsing expression grammars: A recognition-based syntactic foundation. In: Jones, N.D., Leroy, X. (eds.) Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004. pp. 111–122. ACM, Venice, Italy (14–16 January 2004)
6. Hill, O.: Support for Left-Recursive PEGs (2010), https://github.com/orlandohill/peg-left-recursion
7. Hopgood, F.R.A.: Compiling Techniques. MacDonalds (1969)
8. Laurent, N., Mens, K.: Parsing expression grammars made practical. CoRR abs/1509.02439 (2015), http://arxiv.org/abs/1509.02439
9. Mascarenhas, F., Medeiros, S., Ierusalimschy, R.: On the relation between context-free grammars and Parsing Expression Grammars. Science of Computer Programming 89, 235–250 (2014)
10. McClure, R.M.: TMG – a syntax directed compiler. In: Winner, L. (ed.) Proceedings of the 20th ACM National Conference. pp. 262–274. ACM (24–26 August 1965)
11. Medeiros, S.: Correspondência entre PEGs e Classes de Gramáticas Livres de Contexto. Ph.D. thesis, Pontifícia Universidade Católica do Rio de Janeiro (Aug 2010)
12. Medeiros, S., Mascarenhas, F., Ierusalimschy, R.: Left recursion in Parsing Expression Grammars. Science of Computer Programming 96, 177–190 (2014)
13. Redziejowski, R.R.: From EBNF to PEG. Fundamenta Informaticae 128, 177–191 (2013)
14. Redziejowski, R.R.: Trying to understand PEG. Fundamenta Informaticae 157, 463–475 (2018)
15. Sigaud, P.: Left recursion. Tech. rep. (2017), https://github.com/PhilippeSigaud/Pegged/wiki/Left-Recursion
16. Tratt, L.: Direct left-recursive parsing expression grammars. Tech. Rep. EIS-10-01, School of Engineering and Information Sciences, Middlesex University (Oct 2010)
17. Warth, A., Douglass, J.R., Millstein, T.D.: Packrat parsers can support left recursion. In: Proceedings of the 2008 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation, PEPM 2008, San Francisco, California, USA, January 7-8, 2008. pp. 103–110 (2008)

Improving Efficiency of Model Checking for Variants of Alternating-time Temporal Logic (Extended Abstract)

Wojciech Penczek[1,2]

[1] Institute of Computer Science, PAS, Warsaw, Poland
[2] University of Natural Sciences and Humanities, ICS, Siedlce, Poland
penczek@ipipan.waw.pl

1 Multi-agent Systems and ATL*

Multi-agent systems describe interactions of multiple entities called *agents*, often assumed to be intelligent and autonomous [1, 14]. *Alternating-time temporal logic* (ATL*) and its fragment ATL [2] are logics which allow for reasoning about strategic interactions in such systems, by extending the framework of temporal logic with the game-theoretic notion of *strategic ability*. Hence, ATL* enables to express statements about what agents or their groups can achieve. Such properties can be useful for specification, verification, and reasoning about interaction in agent systems [12, 13], as well as about security and usability in e-voting protocols [4, 9]. They have become especially relevant due to active development of algorithms and tools for verification [16], where the "correctness" property is given in terms of strategic ability. While model checking of ATL under perfect information seems to be feasible in practice [5], model checking of ATL under imperfect information [17] is still applicable only to small and medium size systems [10]. This lecture is about selected approaches which can make model checking ATL*, ATL and its time extension TATL more efficient.

2 Model Reduction Methods for Variants of ATL*

Abstraction is a method which typically transforms large (or infinite) models into smaller (or finite) ones, but frequently defined over lattices of more that two *truth* values. We present *multi-valued* ATL* (mv-ATL$^\star_{\preccurlyeq}$), an expressive logic to specify strategic abilities in multi-agent systems [7]. We show how to identify constraints on mv-ATL$^\star_{\preccurlyeq}$ formulas for which the general method for model-independent translation from multi-valued to two-valued model, can be suitably adapted to mv-ATL$^\star_{\preccurlyeq}$, Moreover, we present a model-dependent reduction that can be applied to all formulas of mv-ATL$^\star_{\preccurlyeq}$. In all cases, the complexity of verification increases only polynomially when new truth values are added to the evaluation domain.

Partial order reduction (POR) is another method used to alleviate the state space explosion in model checking [15]. We define a general semantics for strategic abilities of agents in asynchronous systems, with and without perfect information, and present some general complexity results for verification of strategic abilities in asynchronous

systems [11]. A methodology for *POR* in verification of agents with imperfect information is discussed, based on the notion of *traces* introduced by Mazurkiewicz. We define the logic *simple* ATL*, which is the restriction of ATL* such that the strategic modalities cannot be nested and the next step modality is not allowed. Two semantics of *simple* ATL* are considered and it is shown that for memoryless imperfect information contrary to memoryless perfect information, one can apply the partial order reduction techniques known for Linear-time Temporal Logic without the next step operator.

3 Timed ATL

Finally, we discuss Timed Alternating-time Temporal Logic (TATL), a discrete-time extension of ATL. A new semantics, based on counting the number of visits in locations of the history, is introduced in addition to timed memoryful and memoryless ones [3]. We show that all the defined semantics are equivalent for TATL$_{\leq,\geq}$, i.e., when $=$ is not allowed in the formulas. We provide a strategy analysis revealing that it suffices to consider only two actions per location to verify any TATL$_{\leq,\geq}$ formula. This does not extend to TATL. The above results allow for building a hierarchy of strategies comparing the expressive power of the logics against ATL. We discuss a possible impact of this hierarchy on improving efficiency of model checking for TATL$_{\leq,\geq}$.

References

1. T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge. Knowledge and ability. In H.P. van Ditmarsch, J.Y. Halpern, W. van der Hoek, and B.P. Kooi, editors, *Handbook of Epistemic Logic*, pages 543–589. College Publications, 2015.
2. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.
3. É. André, L. Petrucci, W. Jamroga, M. Knapik, and W. Penczek. Timed ATL: forget memory just count. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, pages 1460–1462. IFAAMAS, 2017.
4. F. Belardinelli, R. Condurache, C. Dima, W. Jamroga, and A.V. Jones. Bisimulations for verification of strategic abilities with application to ThreeBallot voting protocol. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, pages 1286–1295. IFAAMAS, 2017.
5. N. Bulling, J. Dix, and W. Jamroga. Model checking logics of strategic ability: Complexity. In M. Dastani, K. Hindriks, and J.-J. Meyer, editors, *Specification and Verification of Multi-Agent Systems*, pages 125–159. Springer, 2010.
6. E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
7. W. Jamroga, B. Konikowska, and W. Penczek. Multi-valued verification of strategic ability. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, pages 1180–1189. IFAAMAS, 2016.
8. W. Jamroga, M. Knapik, and D. Kurpiewski. Fixpoint approximation of strategic abilities under imperfect information. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, pages 1241–1249. IFAAMAS, 2017.
9. W. Jamroga, M. Knapik, and D. Kurpiewski. Model checking the Selene e-voting protocol in multi-agent logics. In *Proceedings of the 3rd International Joint Conference on Electronic Voting (E-VOTE-ID 2018)*, Lecture Notes in Computer Science. Springer, 2018. To appear.

10. W. Jamroga, M. Knapik, D. Kurpiewski, and L. Mikulski. Approximate verification of strategic abilities under imperfect information. *Artificial Intelligence*, 2018. To appear.
11. W. Jamroga, W. Penczek, P. Dembiński, and A. Mazurkiewicz. Towards partial order reductions for strategic ability. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)*, pages 156–165. IFAAMAS, 2018.
12. M. Kacprzak and W. Penczek. Unbounded model checking for Alternating-time Temporal Logic. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 646–653, 2004.
13. M. Kacprzak and W. Penczek. Fully symbolic unbounded model checking for Alternating-time Temporal Logic. *Autonomous Agents and Multi-Agent Systems* 11 (1), 69–89, 2005.
14. A. Lomuscio and W. Penczek. Model checking temporal epistemic logic. In H.P. van Ditmarsch, J.Y. Halpern, W. van der Hoek, and B.P. Kooi, editors, *Handbook of Epistemic Logic*, pages 397–442. College Publications, 2015.
15. A. Lomuscio, W. Penczek, and H. Qu. Partial order reductions for model checking temporal-epistemic logics over interleaved multi-agent systems. *Fundamenta Informaticae*, 101(1-2):71–90, 2010.
16. A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: An open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, 17(1):9–30, 2017. Availabe online.
17. P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.

Linking Exploration Systems with Local Logics over Information Systems

Soma Dutta[1,2], Grzegorz Rozenberg[3,4], and Andrzej Skowron[5,6]

[1] Vistula University
Stokłosy 3, 02-787 Warsaw, Poland
somadutta9@gmail.com
[2] University of Warmia and Mazury
Olsztyn, Poland
[3] Leiden Institute of Advanced Computer Science, Leiden University
P.O. Box 9512, NL-2300 RA Leiden, The Netherlands
[4] Department of Computer Science, University of Colorado at Boulder, USA
430 UCB, Boulder, CO 80309, USA
[5] University of Warsaw, Poland
skowron@mimuw.edu.pl
[6] Systems Research Institute, Polish Academy of Sciences
Newelska 6, 01-447 Warsaw, Poland

Abstract. In the paper, we discuss an extension of exploration systems introduced by Andrzej Ehrenfeucht and Grzegorz Rozenberg. The extension is defined by adding an interpretation of nodes and edges in zoom structure of exploration system. The interpretation is based on the concepts, namely local logic and logic infomorphism, from the notion of information flow by Jon Barwise and Jerry Seligman. This extension makes it possible, in particular, to give a natural interpretation of reaction systems in exploration systems as tools for controlling attention in reasoning about the perceived situation in the physical world.

Key words: reaction system, zoom structure, exploration system, information system, local logic, infomorphism, logic infomorphism

1 Introduction

In the paper, we present a preliminary discussion about possible links between the exploration systems and the information flow approach.

The original motivation behind reaction systems (mostly taken from [1] and [2]) was to model interactions between biochemical reactions in the living cells. Therefore, the formal notion of reaction reflects the basic intuition behind biochemical reactions. A biochemical reaction can take place if in a given state all of its reactants are present and none of its inhibitors is present. When a reaction takes place, it creates its products.

Zoom structures were introduced to integrate structure of a depository of knowledge of a discipline of science (*e.g.*, biology) in the context of reasoning

about the perceived situation related to reaction systems in the physical world. A discipline of knowledge must be structured and the integrating structure here is a well-founded partial order which is well suited to represent a hierarchical structure of knowledge. Exploration systems combine the zoom structures with the reaction systems that are "running within" zoom structures (see, *e.g.*, [3, 4]).

We propose to extend exploration systems by adding interpretation of nodes and edges of zoom structures. The interpretation of nodes of zoom structures, in the form of labels of nodes, is defined by local logics (related to information systems) [5]; the labels of edges in the zoom structure are interpreted as logic infomorphisms between local logics labeling nodes linked by edges. Through local logic it is possible to address a notion of reasoning with respect to local knowledge. Logic infomprphisms can be treated as abstract representations of communications between local logics because each of two local logics linked by a logic infomorphism has some knowledge about facts derivable by the second one. It is possible to treat exploration system as a distritbuted basis for reasoning about the perceived situation related to biochemical processes running in the physical world.

The content of the paper is organized as follows. In Sect. 2 we present the basic concepts of reaction systems. Rudiments of the information flow approach are included in Sect. 3. The zoom structures, exploration systems, and their extension are discussed in Sect. 4.1.

2 Reaction Systems

In this section we recall some basic notions concerning reaction systems (mostly taken from [1] and [2]). The original motivation behind reaction systems was to model interactions between biochemical reactions in the living cells. This leads to the following definitions.

Definition 1. *A* reaction *is a triplet $a = (R_a, I_a, P_a)$, where R_a, I_a, P_a are finite nonempty sets with $R_a \cap I_a = \emptyset$. If S is a set such that $R_a, I_a, P_a \subseteq S$, then a is a reaction in S.*

The sets R_a, I_a, P_a, are called the reactant set of a, the inhibitor set of a, and the product set of a, respectively. Clearly, since R_a, I_a are disjoint and nonempty, then if a is a reaction over S, then $|S| \geq 2$. We will use $rac(S)$ to denote the set of all reactions over S.

The enabling of a (biochemical) reaction in the given state of a biochemical system and the resulting state transformation are defined as follows.

Definition 2. *Let T be a finite set*

- *Let a be a reaction. Then a is* enabled by T*, denoted by $en_a(T)$, if $R_a \subseteq T$ and $I_a \cap T = \emptyset$. The result of a on T, denoted by $res_a(T)$, is defined by: $res_a(T) = P_a$ if $en_a(T)$ and $res_a(T) = \emptyset$, otherwise.*
- *Let A be a finite set of reactions. The* result of A *on T, denoted by $res_A(T)$, is defined by: $res_A(T) = \bigcup_{a \in A} res_a(T)$.*

The intuition behind a finite set T is that of a state of a biochemical system, *i.e.*, a set of biochemical entities present in the current biochemical environment. Thus a single reaction a is enabled by state T if T separates R_a from I_a, *i.e.*, $R_a \subseteq T$ and $I_a \cap T = \emptyset$. When a is enabled by T, then its result on T is just P_a. For a set A of reactions, its result on T is cumulative, *i.e.*, it is the union of the results of all individual reactions from A. Since reactions which are not enabled by T do not contribute to the result of A on T, $res_A(T)$ can be defined by

$$res_A(T) = \bigcup \{res_a(T) | a \in A \text{ and } en_a(T)\}.$$

Now the central notion of a reaction system is defined as follows.

Definition 3. *A* reaction system *is an ordered pair $\mathcal{A} = (S, A)$, where S is a finite set such that $|S| \geq 2$ and $A \subseteq rac(S)$ is a nonempty set of reactions in S.*

Thus a reaction system is basically a finite set of reactions over a set S, which is called the *background set of* $\mathcal{A}$ and its elements are called *entities*. The *result function* of $\mathcal{A}$, $res_{\mathcal{A}} : 2^S \longrightarrow 2^S$ is defined by $res_{\mathcal{A}} = res_A$.

The behaviour of a reaction system (which results from the interactions between its reactions) is determined by its dynamic processes which are formally defined as follows.

Definition 4. *Let $\mathcal{A} = (S, A)$ be a reaction system and let $n \geq 1$ be an integer. An (n-step)* interactive process in A *is a pair $\pi = (\gamma, \delta)$ of finite sequences such that $\gamma = C_0, \ldots, C_n$ and $\delta = D_0, \ldots, D_n$, where $C_0, \ldots, C_n, D_0, \ldots, D_n \subseteq S$, and $D_i = res_{\mathcal{A}}(D_{i-1} \cup C_{i-1})$ for all $i \in \{1, \ldots, n\}$.*

The sequence γ is the *context sequence of* π and the sequence δ is the *result sequence of* π. Then, the sequence $\tau = W_0, W_1, \ldots, W_n$ defined by $W_i = C_i \cup D_i$ for all $i \in \{0, \ldots, n\}$ is the *state sequence of* π with $W_0 = C_0$ called the *initial state of* π (and of τ). If $C_i \subseteq D_i$ for all $i \in \{1, \ldots, n\}$, then we say that π (and τ) is context-independent. Note that we can assume then that $C_i = \emptyset$ for all $i \in \{1, \ldots, n\}$ without changing the state sequence.

Thus, an interactive process begins in the initial state $W_0 = C_0 \cup D_0$. The reactions from A enabled by W_0 produce the result D_1 which together with C_1 forms the successor state $W_1 = C_1 \cup D_1$. The iteration of this procedure determines π: for each $i \in \{0, \ldots, n-1\}$, the successor of state W_i is $W_{i+1} = C_{i+1} \cup D_{i+1}$, where $D_{i+1} = res_{\mathcal{A}}(W_i)$.

The context sequence formalizes the intuition that, in general, a reaction system is not a closed system and so its behavior is influenced by its environment. Note that a context-independent state sequence is determined by its initial state W_0 and the number of steps (n). In general, for an n-step interactive process π of $\mathcal{A}$, π is determined by its context sequence and n.

Also, in a context-independent state sequence $\tau = W_0, \ldots, W_i, W_{i+1}, \ldots, W_n$, during the transition from W_i to W_{i+1} all entities from $W_i - res_{\mathcal{A}}(W_i)$ vanish. This reflects the assumption of *no permanency*: an entity from a current state vanishes unless it is produced/sustained by A. Clearly, if π is not context-independent, then an entity from a current state W_i can be also sustained

(thrown in) by the context (C_{i+1}). This feature is also a major difference with standard models of concurrent systems such as Petri nets (see, *e.g.*, [6]).

3 Barwise and Seligman's logic for distributed system

In [5], the formal counterpart of information available to different sources/agents, including their prior knowledge, is captured through the notion of classification; a classification specifies an agent's information and knowledge regarding which object satisfies which properties or is of which type. The formal definition is given as follows.

Definition 5. *A classification* $A = \langle Tok(A), Typ(A), \models_A \rangle$ *consists of*
(i) a set, $Tok(A)$*, of objects to be classified, called tokens of* A*,*
(ii) a set, $Typ(A)$*, of properties used to classify the tokens, called the types of* A*, and*
(iii) a binary relation, $\models_A$*, between* $Tok(A)$ *and* $Typ(A)$*.*

If $a \models_A \alpha$, then a is said to be of type α in A. That is, $\models_A$ basically specifies which token is of which type. Following the literature of rough sets [7, 8], the notion of classification, presented in [5], can be viewed as a special kind of information system, which is a tuple $(U, \mathcal{A}, \{V_a\}_{a\in\mathcal{A}}, \{f_a\}_{a\in\mathcal{A}})$ consisting of respectively sets of objects, attributes, a set of values for each attribute, and a set of functions for each attribute specifying which object satisfies which attribute with what value. In the context of classification, U is basically $Tok(A)$, $\{(a, v) : a \in \mathcal{A}, v \in V_a\}$ is $Typ(A)$, and for $u \in U$, $f_a(u) = v$ can be associated with $u \models_A (a, v)$ for each $(a, v) \in Typ(A)$.

Now the notion of infomorphism, defined below, represents relationship between classifications, and provides a way of moving information back and forth between them.

Definition 6. *Let* $A = \langle Tok(A), Typ(A), \models_A \rangle$ *and* $B = \langle Tok(B), Typ(B), \models_B \rangle$ *be two classifications. An infomorphism* $f : A \rightleftarrows B$ *from* A *to* B *is a contravariant pair of functions* $f = (\hat{f}, \check{f})$ *such that* $\hat{f} : Typ(A) \mapsto Typ(B)$ *and* $\check{f} : Tok(B) \mapsto Tok(A)$ *satisfying the following fundamental property of infomorphisms.*
$\check{f}(b) \models_A \alpha$ *iff* $b \models_B \hat{f}(\alpha)$ *for each* $b \in Tok(B)$ *and* $\alpha \in Typ(A)$*.*

The notion of an *interpretation*, sometimes also called a *translation* of one language into another, is an example of infomorphism between classifications. There are two aspects of an interpretation; one is to do with tokens (structures), and the other is to do with types (sentences). An interpretation $\mathcal{I} : L_1 \rightleftarrows L_2$ of languages L_1 into L_2 does two things. At the level of types, it associates with every sentence α of L_1, a sentence $\mathcal{I}(\alpha)$ of L_2, its translation. At the level of tokens, it associates with every structure M for L_2, a structure $\mathcal{I}(M)$ for L_1. The relation that $\mathcal{I}(M) \models_{L_1} \alpha$ iff $M \models_{L_2} \mathcal{I}(\alpha)$, presents that what $\mathcal{I}(\alpha)$ says about the structure M is equivalent to what α says about the structure $\mathcal{I}(M)$.

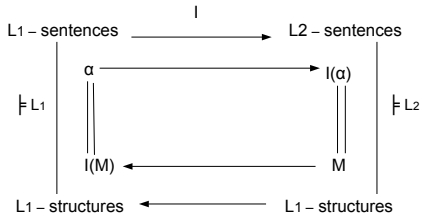


Fig. 1. Interpretation: a translation of one language to another

Definition 7. *Given the infomorphisms $f : A \rightleftarrows B$ and $g : B \rightleftarrows C$, the composition*
$gf : A \rightleftarrows C$ of f and g is the infomorphism defined by $\hat{gf} = \hat{g}\hat{f}$ and $\check{gf} = \check{f}\check{g}$.

Given a classification of information, often it is found that some tokens are identical with respect to some types, and distinct with respect to the rest. The example, as given in [5], might render a better understanding in this regard.

> *My copy of today's edition of the local newspaper bears much in common with that of my next door neighbour. If mine has a picture of President Clinton on page 2, so does hers. If mine has three sections, so does hers. . . . Mine has orange juice spilled on it, hers does not. Hers has the crossword puzzle solved, mine does not.*

In the theory of classification, this aspect is captured by the following notions of invariant and quotient classification.

Definition 8. *Given a classification A, an invariant is a pair $I = (\Sigma, R)$ consisting of a set $\Sigma \subseteq Typ(A)$ of types of A and a binary relation R between tokens of A such that if aRb, then for each $\alpha \in \Sigma$, $a \models_A \alpha$ if and only if $b \models_A \alpha$.*

In the above definition though R needs not to be an equivalence relation, in the further considerations R is considered to be the smallest equivalence relation containing the concerned relation.

Definition 9. *Let $I = (\Sigma, R)$ be an invariant on the classification A with respect to an equivalence relation R. The quotient of A by I, denoted as A/I, is the classification with types Σ, whose tokens are the R-equivalence classes of tokens of A, and with $[a]_R \models_{A/I} \alpha$ if and only if $a \models_A \alpha$.*

One can notice that the notion of invariance, as defined in [5], also corresponds to the notion of indiscernibility in the context of rough set literature. In an information system, given by the tuple $(U, \mathcal{A}, \{V_a\}_{a\in\mathcal{A}}, \{f_a\}_{a\in\mathcal{A}})$, two objects x, y of U are said to be indiscernible (i.e., $xIND(A)y$) if $f_a(x)$ and $f_a(y)$ receive the same value from V_a for any $a \in \mathcal{A}$. Moreover, the notion of sequent, defined below, also has a counterpart in rough set literature. A sequent can be viewed as a non-deterministic decision rule, i.e., relation between two (finite) sets of

descriptors (e.g. (a, v) for $a \in \mathcal{A}$ and $v \in V_a$) describing the available data of the information system.

Example 1. For a given information system $\mathbb{A}=(U, \mathcal{A}, \{V_a\}_{a\in\mathcal{A}}, \{f_a\}_{a\in\mathcal{A}})$ and the indiscernibility relation $IND(A)$ one can define two classifications $Cl(\mathbb{A}) = (U, \Sigma, \models_{\mathbb{A}})$ and $Cl(\mathbb{A}/IND(A)) = (U/IND(A), \Sigma, \models_{\mathbb{A}/IND(A)})$, where Σ is a subset of $Type(A)$ (cf. below Def. 5), $x \models_{\mathbb{A}} \alpha$ denotes that x satisfies α, and $[x]_{IND(A)} \models_{\mathbb{A}/IND(A)} \alpha$ means that $x \models_{\mathbb{A}} \alpha$ [7, 8]. One can easily check that these two classifications can be linked by infomorphisms $(id, g) : Cl(\mathbb{A}) \rightleftarrows Cl(\mathbb{A}/IND(A))$, where id is the identity on Σ and g assigns to $[x]_{IND(A)}$ any object from $[x]_{IND(A)}$ and $(id, h) : Cl(\mathbb{A}/IND(A)) \rightleftarrows Cl(\mathbb{A})$, where id is the identity on Σ and $h(x) = [x]_{IND(A)}$ for $x \in U$.

□

As pointed out in [5],

> *one way to think about information flow in a distributed system is in terms of a 'theory' of the system, that is, a set of known laws that describes the system.*

Based on this general notion of classification, the notion of sequent or notion of consequence of a deductive logic is captured as follows.

As a classification, say $(Tok(A), Typ(A), \models_A)$, specifies a perspective about the properties of the $Tok(A)$ we may call the classification as *classification of A* considering A to refer to that particular perspective.

Definition 10. *Let $cl(A) = (Tok(A), Typ(A), \models_A)$ be a classification of A.*
(i) For any $\Gamma, \Delta \subseteq Typ(A)$, $\langle \Gamma, \Delta \rangle$ is considered to be a sequent of $Typ(A)$.
(ii) A sequent $\langle \Gamma, \Delta \rangle$ is a partition of $\Sigma' \subseteq Typ(A)$ if $\Gamma \cup \Delta = \Sigma'$ and $\Gamma \cap \Delta = \phi$.
(iii) A binary relation $\vdash$ between subsets of $Typ(A)$ is called a (Gentzen) consequence relation.
(iv) A theory $T = (\Sigma, \vdash)$ is a pair, where $\Sigma \subseteq Typ(A)$ and $\vdash$ is a consequence relation on Σ.
(v) A constraint of the theory T is a sequent $\langle \Gamma, \Delta \rangle$ such that $\Gamma \vdash \Delta$.
(vi) A token a of $Tok(A)$ satisfies $\langle \Gamma, \Delta \rangle$ provided that if a is of type α for every $\alpha \in \Gamma$, then a is of type β for some $\beta \in \Delta$. A token not satisfying a sequent is called a counterexample to the sequent.
(vii) The theory $T(cl(A)) = (Typ(A), \vdash_A)$ generated by $cl(A)$ is the theory whose constraints are the set of sequents satisfied by every token of $Tok(A)$.
(viii) A theory whose constraints are satisfied by every token of the classification is called a complete theory.

Here it is to be noted that sequents are all possible pairs of sets of types, and some of them come under the consequence relation. Usually, some natural conditions are imposed on the set of sequents if one would like to consider it as a theory. Below we present such conditions in the definition of the regular theory.

Definition 11. *A theory $T = (\Sigma, \vdash)$ is regular if it satisfies the following properties viz., identity, weakening, and global cut for all types α, and all set $\Gamma, \Gamma', \Delta, \Delta', \Sigma', \Sigma_0, \Sigma_1$ of types.*
Identity $\alpha \vdash \alpha$
Weakening *If $\Gamma \vdash \Delta$, then $\Gamma, \Gamma' \vdash \Delta, \Delta'$.*
Global cut *If $\Gamma, \Sigma_0 \vdash \Delta, \Sigma_1$ for each partition $\langle \Sigma_0, \Sigma_1 \rangle$ of Σ', then $\Gamma \vdash \Delta$.*

Proposition 1. *The theory $T(cl(A)) = (Typ(A), \vdash_A)$ generated by the classification $cl(A)$ of A is a regular theory.*

Proposition 2. *Any regular theory $T = (\Sigma, \vdash)$ satisfies the following condition.*
Finite cut: If $\Gamma, \alpha \vdash \Delta$ and $\Gamma \vdash \Delta, \alpha$, then $\Gamma \vdash \Delta$.

Definition 12. *Given two theories $T_1 = (Typ(T_1), \vdash_{T_1})$ and $T_2 = (Typ(T_2), \vdash_{T_2})$, a (regular theory) interpretation $f : T_1 \mapsto T_2$ is a function from $Typ(T_1)$ to $Typ(T_2)$ such that for each $\Gamma, \Delta \subseteq Typ(T_1)$ if $\Gamma \vdash_{T_1} \Delta$, then $f(\Gamma) \vdash_{T_2} f(\Delta)$.*

The notion of local logic puts the idea of a classification together with that of a regular theory. Moreover, introducing a notion of normal tokens it models resonable but unsound inferences.

Definition 13. *A local logic $\mathcal{L} = (Tok(\mathcal{L}), Typ(\mathcal{L}), \models_{\mathcal{L}}, \vdash_{\mathcal{L}}, N_{\mathcal{L}})$ consists of*
(i) a classification $cl(\mathcal{L}) = (Tok(\mathcal{L}), Typ(\mathcal{L}), \models_{\mathcal{L}})$,
(ii) a regular theory $Th(\mathcal{L}) = (Typ(\mathcal{L}), \vdash_{\mathcal{L}})$, and
(iii) a subset $N_{\mathcal{L}} \subseteq Tok(\mathcal{L})$, called the normal tokens of $\mathcal{L}$, which satisfies all the constraints of $Th(\mathcal{L})$.

Definition 14. *A logic infomormhism $f : \mathcal{L}_1 \rightleftarrows \mathcal{L}_2$ consists of a contravariant pair $f = (\hat{f}, \check{f})$ of functions such that*
(i) $f : cl(\mathcal{L}_1) \rightleftarrows cl(\mathcal{L}_2)$ is an infomorphism of classifications,
(ii) $\hat{f} : Th(\mathcal{L}_1) \mapsto Th(\mathcal{L}_2)$ is a theory interpretation, and
(iii) $\check{f}(N_{\mathcal{L}_2}) \subseteq N_{\mathcal{L}_1}$.

It can be observed that through these notions of classification, local logic, and logic infomorphism the target of the authors [5] was to formalize respectively an individual's information base, logical reasoning base, and flow of information from one individual to another in the process of decision making.

4 Exploration Systems and Their Extension Grounded in Local Logics over Information Systems

In this section, we consider exploration systems which combine zoom structures with reaction systems "running within" zoom structures (see, *e.g.*, [3, 4]). The original intuition and motivation was that a zoom structure is the integrating structure of a depository of knowledge of a discipline of science (*e.g.*, biology). A discipline of knowledge must be structured and the integrating structure here

is a well-founded partial order which is well suited to represent a hierarchical structure of knowledge (as, *e.g.*, is the case in biology).

Formally zoom structures are defined as follows (we consider irreflexive partial orders; recall that a partial order is well-founded if every walk against its edges is finite).

Definition 15. *A* zoom structure *is a 6-tuple* $\mathcal{Z} = (D, E, \Gamma, \Delta, \{D_i\}_{i\in\Gamma}, \{E_j\}_{j\in\Delta})$, *where*

(i) D *is a non-empty set,*
(ii) $E \subseteq D \times D$ *is such that the* E^+ *(*i.e.*, the transitive closure of* E*) is a well-founded partial order,*
(iii) Γ, Δ *are finite sets,*
(iv) $\{D_i\}_{i\in\Gamma}$ *is a partition of* D *(into non-empty sets), and*
(v) $\{E_j\}_{j\in\Delta}$ *is a partition of* E *(into non-empty sets).*

Obviously, $\mathcal{Z}$ can be also seen as a node- and edge-labelled graph, where D is its set of nodes labelled by elements of Γ, and E is its set of edges labelled by elements of Δ.

Data structures for implementing large sets of data are often hierarchical: in accessing specific data one usually performs a series of zoom operations each of which leads from a topic to its "subtopic." This is reflected in the basic notion of an inzoom of $\mathcal{Z}$.

Definition 16. *Let* $\mathcal{Z} = (D, E, \Gamma, \Delta, \{D_i\}_{i\in\Gamma}, \{E_j\}_{j\in\Delta})$ *be a zoom structure. An* inzoom *of* $\mathcal{Z}$ *is a finite sequence* $x = x_1, x_2, \ldots, x_n$ *such that* $n \geq 2$, $x_i \in D$ *for* $i \in \{1, \ldots, n\}$, *and, for each* $i \in \{2, \ldots, n\}$, $(x_i, x_{i-1}) \in E$.

The set of inzooms of $\mathcal{Z}$ is denoted by $INZOOM(\mathcal{Z})$.

Thus an inzoom represents a "reverse walk" in $\mathcal{Z}$, *i.e.*, a walk through nodes such that each single step goes against an edge of E. In the framework of zoom structures, inzooms (rather than nodes) are basic units for reasoning about and the usage of zoom structures.

While a zoom structure represents the *static* integrating structure of a depository of knowledge, the dynamic processes of exploring depositories of knowledge are represented by reaction systems "embedded" (rooted) in zoom structures. The embedding of a reaction system in a zoom structure is realized by requiring that the background of the reaction system consists of inzooms of the zoom structure.

Definition 17. *Let* $\mathcal{Z}$ *be a zoom structure. A reaction system* $\mathcal{A} = (S, A)$ *is* rooted in $\mathcal{Z}$ *if* $S \subseteq INZOOM(\mathcal{Z})$.

Recall that a reaction system $\mathcal{A} = (S, A)$ specifies, through the result function $res_{\mathcal{A}}$, a set-theoretical transformation of the set of subsets of its background set S (hence on the states of $\mathcal{A}$). (When one allows processes of $\mathcal{A}$ to be more general than context-independent, then more general transformations are considered.) The background set can be any set and if we choose it to be a set of zooms of

$\mathcal{Z}$, then we root $\mathcal{A}$ in $\mathcal{Z}$, " allowing" $\mathcal{A}$ to explore (the knowledge deposited in) $\mathcal{Z}$.

This leads to the notion of an exploration system.

Definition 18. *An* exploration system *is an ordered pair* $\mathcal{E} = (\mathcal{Z}, \mathcal{F})$, *where* $\mathcal{Z}$ *is an extended zoom structure and* $\mathcal{F}$ *is a family of reaction systems rooted in* $\mathcal{Z}$.

In the original definition (see [3]) $\mathcal{Z}$ is a construct more general than a zoom structure. However, for the purpose of our discussion it suffices to assume here that $\mathcal{Z}$ is a zoom structure.

Exploration systems can be used for reasoning about perceived situation in the physical world. Note that objects in D do not have to belong to the ground level of hierarchical modeling obtained by sensory based perception of reality. They can be constructs of higher level of the hierarchical modeling for perception based reasoning about the currently perceived situation. Moreover, edges in E can be interpreted as links representing possible relevant interactions between objects from D. This means that results of interactions can be used in perception based reasoning about the currently perceived situation.

Example 2. Let $\mathcal{Z} = (D, E, \Gamma, \Delta, \{D_i\}_{i\in\Gamma}, \{E_j\}_{j\in\Delta})$ be a zoom structure such that $D = \{x_1, x_2, \ldots, x_{10}\}$, $\Gamma = \{1, 2, 3\}$, $\Delta = \{4, 5, 6\}$, $D_1 = \{x_1, x_2, x_3\}$, $D_2 = \{x_4, x_5, x_6, x_7\}$, $D_3 = \{x_8, x_9, x_{10}\}$, $E = E_4 \cup E_5 \cup E_6$, $E_4 = \{(x_5, x_7), (x_8, x_{10}), (x_1, x_4), (x_3, x_5)\}$, $E_5 = \{(x_6, x_7), (x_4, x_7), (x_2, x_3)\}$, and $E_6 = \{(x_5, x_{10}), (x_1, x_3), (x_1, x_2), (x_3, x_8), (x_9, x_{10})\}$ (see Figure 2). It is illustrated in Figure 2.

Let $\mathcal{A} = (S, A)$ be a reaction system rooted in $\mathcal{Z}$ such that $S = \{(x_3, x_2, x_1), (x_3, x_1), (x_3, x_2), (x_2, x_1), (x_7, x_4), (x_{10}, x_8)\}$ and A contains the reaction $a = (R_a, I_a, P_a)$ with $R_a = \{(x_3, x_2, x_1), (x_2, x_1)\}$, $I_a = \{(x_{10}, x_8)\}$, and $P_a = \{(x_7, x_4)\}$. This gives an example of a reaction system rooted in a zoom structure.

4.1 Exploration Systems Grounded in the Space of Information Systems

We propose to extend exploration systems by adding interpretation of nodes and edges of zoom structures. The interpretation of nodes of zoom structures is given in the form of labels of nodes defined by local logics (related to information systems) [5] and interpretation of edges in the zoom structure are logic infomorphisms between local logics labeling nodes linked by the edges.

Having such a framework, following the information flow approach [5] one can construct local logics for individual agents as well as local logic representing the whole network of local logics. However, such a global logic will be very complex what makes it hardly possible to derive efficiently conclusions of the basis of such a local logic. Moreover, due to the cumulation of uncertainties the reasoning on the basis of such a logic may be not satisfactory.

Instead of this we propose to construct local logics only for some fragments of zoom structures which are relevant for the perceived situation. Namely, we propose to construct local logics corresponding to subnetworks defined by the

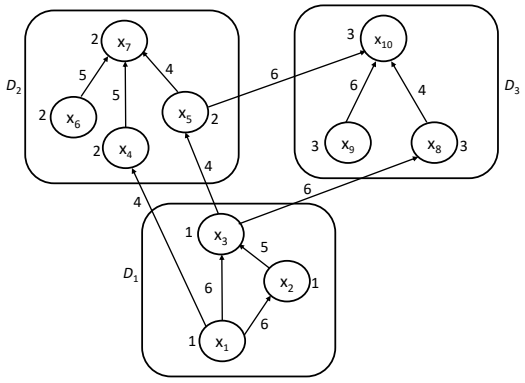


Fig. 2. Zoom structure from Example 2.

partition of nodes given by a zoom structure (representing subdomains of knowledge). It should be noted that the partition blocks can be further restricted by using reactants of relevant reactions from exploration system. In the consequence, the fragments of networks for which it is necessary to construct locals logic representing them is substantially reduced. The aim is to make the reasoning process efficient and leading to conclusions on the perceived situation. The products of reactions from the considered exploration system are used as pointers indicating relevant fragments of zoom structure. These fragments are used in further steps of reasoning on the basis of local logics toward understanding the perceived situation.

There is one more extension we propose to the zoom structure defined above. This is specified by a selection function making it possible to select, from the family of reaction systems given in the considered exploration system, a relevant reaction system, for the next step of reasoning on the basis of the current information on the currently perceived situation. We assume here that this information is represented, in particular, in a distinguished nodes (called sensory nodes) of extended zoom structure, where information systems and corresponding to them local logics are labeling nodes.

5 Conclusions

We presented a preliminary discussion about extension of exploration systems defined in [3, 4]. In the full version of the paper we plan to give more details about this extension and its possible applications.

In our further research, we plan to consider the exploration systems as dynamic complex networks with the structures changing by the control mechanisms

responsible for the behavior of exploration systems. The control of an agent, using a given exploration system interacting with the environment, is aiming to satisfy the 'needs' of the agent. It should be noted that the needs may change with time. One may ask how such complex exploration systems may be constructed and modified with time. Here, we would like to point to two special strategies following two kinds of judgments used for making changes in the current exploration system. The first one is based on aggregation of information systems labeling nodes of zoom structures of these exploration systems and consequently the local logics corresponding to them. The aggregations are such as operations of join of information systems with some relevant constraints. These constraints are used to filter Cartesian products of sets of objects in the joint information systems to obtain relevant computational building blocks (granules) for describing the perceived situation, *e.g.*, the ones which are used for approximation of complex vague concepts responsible for triggering action or plans (see, *e.g.*, [9]). The second kind of strategies is based on the ability of agents to create the so called complex granules making it possible to extend the fragments of the physical world, perceived by agents, to the new fragments localized in the scope of these complex granules (see, *e.g.*, [10–12]). More detailed discussion on the issues related to dynamic behavior of exploration systems will be included in our next papers.

References

1. Ehrenfeucht, A., Rozenberg, G.: Reaction systems. Fundamenta Informaticae **76** (2006) 1–18
2. Ehrenfeucht, A., Petre, I., Rozenberg, G.: Reaction systems: A model of computation inspired by the functioning of the living cell. In Konstantinidis, S., Moreira, N., Reis, R., Shallit, J., eds.: The Role of Theory in Computer Science - Essays Dedicated to Janusz Brzozowski, World Scientific (2017) 1–32
3. Ehrenfeucht, A., Rozenberg, G.: Zoom structures and reaction systems yield exploration systems. International Journal of Foundations of Computer Science **25** (2014) 275–306
4. Ehrenfeucht, A., Rozenberg, G.: Standard and ordered zoom structures. Theoretical Computer Science **608** (2015) 4–5
5. Barwise, J., Seligman, J.: Information Flow: The Logic of Distributed Systems. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK (1997)
6. Reisig, W.: Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Springer, Heidelberg (2013)
7. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences **177**(1) (2007) 3–27
8. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Sciences **177**(1) (2007) 28–40
9. Skowron, A., Stepaniuk, J.: Hierarchical modelling in searching for complex patterns: Constrained sums of information systems. Journal of Experimental and Theoretical Artificial Intelligence **17** (2005) 83–102

10. Jankowski, A.: Interactive Granular Computations in Networks and Systems Engineering: A Practical Perspective. Lecture Notes in Networks and Systems. Springer, Heidelberg (2017)
11. Skowron, A., Jankowski, A.: Interactive computations: Toward risk management in interactive intelligent systems. Natural Computing **15(3)** (2016) 465 – 476
12. Skowron, A., Jankowski, A.: Rough sets and interactive granular computing. Fundamenta Informaticae **147** (2016) 371–385

Simulating Gene Regulatory Networks using Reaction Systems

Roberto Barbuti, Pasquale Bove, Roberta Gori,
Francesca Levi, and Paolo Milazzo

Dipartimento di Informatica, Università di Pisa, Italy

Abstract. Gene Regulatory Networks represent the interactions among genes regulating the activation of specific cell functionalities and they have been successfully modeled using threshold Boolean networks. In this paper we propose a systematic translation of threshold Boolean networks into Reaction Systems. Our translation produces a non redundant set of rules with the minimal number of objects. This translation allows us to simulate the behavior of a Boolean network simply by executing the (closed) Reaction System we obtain. Moreover, it allows us to investigate the role of different genes simply by "playing" with the rules related to different genes. Starting from a well-known Boolean network model of the Yeast-Cell Cycle, we construct the corresponding Reaction System and start an investigation on causal dependencies among genes.

1 Introduction

In the context of molecular biology of cells, *Gene Regulatory Networks* (GRNs) represent the interactions among genes regulating the activation of specific cell functionalities. More specifically, genes in a GRN can be either active (i.e. the corresponding protein is expressed) or not, and each active gene can either stimulate or inhibit the activation of a number of other genes. Moreover, the activation of some genes is also usually influenced by other factors such as the availability of some substances in the environment or the reception of a signal form neighbor cells. As a consequence, gene regulatory networks can be seen as the mechanism that allow a cell to react to external stimuli. When a stimulus is received, it causes a change in the activation state of a few genes, that, in turn, influence other genes, allowing a new configuration of active genes (corresponding to a new set of active cell functionalities) to be reached.

Several approaches have been proposed to model and analyze GRNs (see [24] for a survey on this topic). Modeling techniques can either deal only with the qualitative aspects of such networks (treating them essentially as logic circuits), or can describe also the quantitative aspects, such as the rates of the interactions. The latter approach is for sure more precise, but requires many additional details of the network dynamics to be taken into account, such as the rates of transcription and translation of genes' DNA into proteins and the rates of protein-protein and protein-DNA interactions. Qualitative models are often sufficient to reason on the behavior of the regulatory networks, although with some degree of approximation.

In the qualitative modeling setting, one of the most successful modeling frameworks for gene regulatory networks are *Boolean networks*. In this setting, a particular simple form of Boolean networks, the so called *threshold Boolean networks* [16, 19, 22], have been widely used to model the dynamics of quite complex regulatory networks. In threshold Boolean networks, the Boolean function of each node depends on the sum of its input signals only. This variant of Boolean networks can be easily implemented and, at the same time, it is well suited for representing gene regulatory networks.

Boolean networks allow dynamical properties of GRNs to be investigated. Starting from an initial configuration of active genes, the dynamics of a GRNs is expressed as a sequence of steps in which such a configuration is updated according to the influences among the genes described by the Boolean network. Dynamical properties can be investigated either by performing simulations, or by constructing the (finite) graph representing all possible dynamical evolutions. Example of properties that are often studied on these models are reachability and stability of configurations, and confluence of evolutions started from different initial configurations into stable configurations (attractors).

Analysis of dynamical properties may become computationally very expensive. In order to reduce the state space to be analyzed, minimization techniques can be applied. The Boolean function represented by a Boolean network can be synthesized in any framework of logic minimization. The classical approach to logic minimization that produces sum of products two level formulas can be used (see e.g., Espresso [10, 23]). More than two level minimization is harder, but the size of the resulting expressions can significantly decrease. In particular, bounded multilevel forms, such as three or four-level forms [8, 7, 9, 10] could represent a good tradeoff among the cost of the final representation and the time needed for the minimization procedure.

Other analysis methods for GRNs could be applied by changing the representation of the Boolean networks describing them. In this paper we propose a systematic translation of threshold Boolean networks into *Reaction Systems* [17, 13]. Reaction systems were introduced by Ehrenfeucht and Rozenberg as a novel model for the description of biochemical processes driven by the interaction among reactions in living cells. Reaction systems are based on two opposite mechanisms, namely *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 state of a Reaction System consists of a finite set of objects which can evolve by means of application of reactions. The presence of an object in a state expresses the fact that the corresponding biological entity, in the real system being modeled, is present. Quantities (or concentrations) of the entities are not described: Reaction Systems are hence a *qualitative* modeling formalism.

In this setting the dynamic run of the Reaction System simulates the evolution of the Boolean network. This correspondence allows us to "play" with the rules of the Reaction System related to different genes in order to detect dynamic causality dependencies between genes activation/deactivation. Moreover, we believe that this correspondence will allow us to apply to Boolean networks well-known techniques to detect causality relationships between objects in bio-

logical systems. The understanding of causality relationships among the events happening in a biological (or bio-inspired) system is an issue investigated in the context both of systems biology (see e.g. [18, 11, 12]) and of natural computing (see e.g. [15]).

In [14] Brijder, Ehrenfeucht and Rozenberg initiate an investigation of *causalities* in Reaction Systems [17, 13]. Causalities deal with the ways entities of a Reaction System influence each other. In [14], both static/structural causalities and dynamic causalities are discussed, introducing the idea of *predictor*. In [2, 1, 3, 5, 4, 6], the idea of predictors was enhanced by defining the notions of *formula based predictor* and *specialized formula based predictor*. These new concepts allow us to study all causal dependencies of one object from all others. We believe that the Reaction System simulating a Boolean network can then be investigated by computing the specialized formula based predictor of a particular activation/deactivation gene configuration. This will allow us to obtain a logic formula characterizing all alternative activation/deactivation gene configurations that lead to the requested configuration in a bounded number of steps. This could be very useful to understand which genes are *necessary* for reaching a requested configuration.

The translation we propose in this paper produces a *non redundant* set of rules with the *minimal number* of objects.

The paper is organized as follows. Section 2 introduces the main concepts of (Closed) Reaction Systems. In Section 3 we describe how Boolean networks are defined and how they work. Section 4 presents the systematic translation from Boolean network to Reaction Systems we propose. Finally, in Section 5 we apply our methodology to simulate and study the Yeast-Cell Cycle Boolean Network.

2 Closed Reaction Systems

In this section we recall the basic definition of Reaction Systems [17, 13]. 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. Reactants and inhibitors $R \cup I$ of a reaction are collectively called *resources* of such a reaction, and we assume them to be disjoint ($R \cap I = \emptyset$), otherwise the reaction would never be applicable. The set of all possible reactions over a set S is denoted by $\mathrm{rac}(S)$. Finally, a *Reaction System* is a pair $\mathcal{A} = (S, A)$, where S is a finite support set, and $A \subseteq \mathrm{rac}(S)$ is a 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 $\mathrm{res}_a(T)$ 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 ($t$hreshold supply assumption). Therefore, each reaction which is not inhibited can be applied, and the result of the application of multiple reactions is cumulative. Formally, given a Reaction System $\mathcal{A} = (S, A)$, the result of application of $\mathcal{A}$ to a set $T \subseteq S$ is defined as $\mathrm{res}_{\mathcal{A}}(T) = \mathrm{res}_A(T) = \bigcup_{a \in A} \mathrm{res}_a(T)$.

An important feature of Reaction System is the assumption about the *non-permanency* of objects: 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 in any reaction.

The dynamics of a Reaction System is generally driven by the *contextual* objects, namely the objects supplied to the system by the external environment at each step. *Closed* Reaction Systems are the subset of general Reaction Systems where the external environment provides objects at the first step only.

This allows us to simplify the dynamics of a (closed) Reaction System $\mathcal{A} = (S, A)$. Indeed, given the initial set D_0 the semantics can be simply defined as the *result sequence*, $\delta = D_1, \ldots, D_n$ where each set D_i, for $i \geq 1$, is obtained from the application of reactions $\mathcal{A}$ to the state obtained at the previous step D_{i-1} ; formally $D_i = res_{\mathcal{A}}(D_{i-1})$ for all $1 \leq i < n$. For the sake of simplicity, we write $D_{i-1} \rightarrow_{\mathcal{A}} D_i$ as a shorthand for $D_i = res_{\mathcal{A}}(D_{i-1})$. In this case the sequence of states of the Reaction System coincides with the result sequence $\delta = D_1, \ldots, D_n$.

3 Boolean Networks

We present a formal definition of threshold Boolean networks [22] considering a set M of n elements, $S_1, S_2, \ldots S_n$ to be *nodes* of a network. We assign to each element, at each time instant t, a value $S_i(t) \in \{0, 1\}$ denoting if the element S_i is present at that instant or not. The interactions among elements are given by the set of *edges* of the network called E. Each edge in E can be either *activating* or *inhibiting*. An edge from element S_j to element S_i is denoted a_{ij} (where $i \neq j$ given that a element cannot either activate or inhibit itself). An activating edge has value 1 while an inhibiting one has value -1. Elements M can be partitioned in two sets M_{sa} and M_{nsa} of *self-activating* and *non-self-activating* elements, respectively, $M = M_{sa} \cup M_{nsa}$. A self-activating elements if is present at time t and it is not inhibited will be present at time $t + 1$, while a non-self-activating one will not. Moreover we assume that each element S_i has associated a value $\theta_i \in \Theta$ which is called the *threshold parameter* of S_i. The pair (M, E) is called a *threshold Boolean network*.

The states of the nodes in the network are updated in parallel in discrete time. The rules for updating the values of nodes are the following, for $i \in \{1, \ldots, n\}$

$$S_i(t+1) = \begin{cases} 1 & \text{if } \sum_j a_{ij} S_j(t) > \theta_i \\ 0 & \text{if } \sum_j a_{ij} S_j(t) < \theta_i \\ S_i(t) & \text{if } S_i \in M_{sa} \wedge \sum_j a_{ij} S_j(t) = \theta_i \\ 0 & \text{if } S_i \in M_{nsa} \wedge \sum_j a_{ij} S_j(t) = \theta_i \end{cases}$$

where the value θ_i is the threshold parameter associated to the element S_i.

Typically the threshold parameter θ_i associated to S_i is equal to 0 so that the switch is inactive if there is no input signal, and it switches on when signals are present. A node which needs more than one incoming signals to be activated can be represented in the model by setting θ_i to a value greater than 0.

Starting from an initial condition, the network produces a dynamical sequence of states and it can reach a periodic attractor or a fixed point.

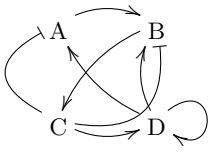


Fig. 1. An example of gene regulatory network.

There is a natural representation of Boolean networks by means of graph where the nodes represent the elements and the edges represent the interactions between the elements; an *activating edge* is indicated by $\rightarrow$ while an *inhibiting* one is indicated by $\dashv$. Non self-activating genes are represented by nodes with half-arrow ($\rightharpoonup$) self loops.

We introduce an example to illustrate threshold Boolean networks and their dynamic evolution.

Example 1. Let us consider the threshold Boolean network (M, E) with elements $M = \{A, B, C, D\}$ such that $M_{sa} = \{A, B, C\}$ and $M_{nsa} = \{D\}$ and with the edges depicted in Fig. 1. Thus, the element D is non self-activating while the elements A, B and C are self-activating. We also assume that the threshold parameter for each element is 0.

We describe the temporal evolution considering an initial state in which the element D is present while the others are not. We have

Step	A	B	C	D
1	0	0	0	1
2	1	1	0	0
3	1	1	1	0
4	0	1	1	1
5	0	1	1	0
6	0	0	1	0
7	0	0	1	0

Initially the element D stimulate the activation of both elements A and B because the element C, their inhibitor is not present. Note that at the second step the element D is inactive because it is non self-activating. Then, at step 3, the elements C is present because it is activated by B while A and B are still present because they are self-activating and at step 2 C was not present. At step 4, the element C actives D and inhibits A which is not present anymore. By contrast, C does not inhibit the element B which is still present because it was present at step 3 but also A was present. The step 5 is similar. Finally at step 6 the element B is not present anymore because in the previous configuration A and D were absent and the inhibitor C was activated. The last one is also a stationary state of the system, since no more evolutions of the network are possible from such state.

4 Translation of Boolean Networks into Reaction Systems

We present a translation of threshold boolean networks in closed Reaction System. Given a Boolean network (M, E) with $M = \{S_1, S_2, \dots S_n\}$ we define for $S_i \in M$,

$$Act(S_i) = \{S_j \mid j \in [1, n] \ \wedge \ a_{ij} = 1\} \qquad In(S_i) = \{S_j \mid j \in [1, n] \ \wedge \ a_{ij} = -1\}$$

We recall that a_{ij} denotes an edge from element S_j to element S_i. Hence, $Act(S_i)$ reports the elements S_j which *activates* S_i and analogously $In(S_i)$ reports the elements S_j which *inhibits* it.

Definition 1. *Let* (M, E) *be a* threshold Boolean network *with elements* $M = \{S_1, S_2, \dots S_n\}$ *and threshold parameters* $\Theta = \{\theta_1, \theta_2, \dots \theta_n\}$. *We define its translation as the closed* Reaction System $\mathcal{RS}((M, E)) = (M, A)$, *where reactions in* A *are constructed according to the following inference rules:*

$$1) \ \frac{P_i \subseteq Act(S_i) \quad I_i \subseteq In(S_i) \quad \#P_i - \#(In(S_i) \setminus I_i) = \theta_i - 1}{(P_i, I_i, \{S_i\}) \in A} \qquad 2) \ \frac{S_i \in M_{sa} \quad P_i \subseteq Act(S_i) \quad I_i \subseteq In(S_i) \quad \#P_i - \#(In(S_i) \setminus I_i) = \theta_i}{(P_i \cup \{S_i\}, I_i, \{S_i\}) \in A}$$

The closed Reaction System $\mathcal{RS}((M, E))$ simulates the threshold Boolean network (M, E) using reactions obtained by applying either the inference rule 1) or the inference rule 2). Rule 1) defines the reactions which simulate the production of a element S_i at time $t + 1$ whenever at time t the number of the elements which activate S_i minus the number of the elements which inhibit S_i is greater than θ_i (according to the rule given in Section 3). This behavior is simulated by a reaction which has as *product* S_i, as *reactants* P_i and as *inhibitors* I_i where P_i is a subset of the elements which activates S_i and analogously I_i is a corresponding subset of the elements which inhibits it. Note that this reaction can be applied if in the Reaction System state none of the elements in I_i is present. As a consequence, the set of the elements which are inhibitors of S_i and which may be present is given by $In(S_i) \setminus I_i$. Therefore we guarantee that the cardinality of P_i minus that of $In(S_i) \setminus I_i$ is greater than θ_i. More specifically we require that this difference is equal to $\theta_i - 1$ in order to built a minimal set of reactions in the corresponding Reaction System.

Rule 2) is defined for self-activating elements which remains active at time $t + 1$ if they are present at time t and they are not inhibited (according to the rule given in Section 3). In Reaction System, due to the non-permanency of objects, the objects carried over to the next step are only those produced by reactions. Therefore, in this case, the reaction which simulates the behavior has S_i as *reagent* and also as *product*. Similarly as in the case 1), P_i is a subset of the elements which activates S_i and I_i is a corresponding subset of the elements which inhibits S_i. In this case, however, we require that the cardinality P_i minus the number of inhibitors which might be present (i.e. $(In(S_i) \setminus I_i)$) is exactly θ_i.

Example 2. We give the translation of the threshold Boolean network (M, E) presented in Example 1. Fig. 1) illustrates the interactions between the elements of the network $M = \{A, B, C, D\}$ such that $M_{sa} = \{A, B, C\}$ and $M_{nsa} = \{D\}$.

By assuming again that the threshold parameter for each element is 0 we obtain the closed Reaction System $\mathcal{RS}((M,E)) = (M,A)$ with reactions A are defined as follows:

$$
\begin{array}{llll}
(\{C\},\emptyset,\{D\}), & (\{B\},\emptyset,\{C\}) & (\{C\},\emptyset,\{C\}), & (\{A,D\},\emptyset,\{A\}) \\
(\{D\},\{C\},\{A\}), & (\{A\},\{C\},\{B\}) & (\{A\},\{C\},\{A\}), & (\{B,D\},\emptyset,\{B\}) \\
(\{D\},\{C\},\{B\}), & (\{A,D\},\emptyset,\{B\}) & (\{B,A\},\emptyset,\{B\}), & (\{B\},\{C\},\{B\})
\end{array}
$$

The reactions on the left are obtained by applying the rule 1) while those on the right by applying the rule 2). The two columns on the left contains one or more reaction for each elements which produces the element. For the elements D and C there is exactly one reaction given that they do not have inhibitors. By contrast, the element C inhibits both A and B. In the first case, there is just one reaction which says that D produces A if not inhibited by C. The case of B is similar but it can be activated by two element, A and D. Hence, there are three different reactions corresponding to the possible conditions of elements which can activate and inhibit B. Note that the requirements of rule 1) guarantee that only minimal subsets are considered. For instance a reaction such as $(\{A,D\},\{C\},\{B\})$ is subsumed by $(\{A,D\},\emptyset,\{B\})$ given that the latter reaction can be applied regardless of the presence of C.

The two columns on the right shows the reactions for self-activating elements, A, B and C. Similarly as in the previous case there is one or more reaction for each self-activating elements which has the element both as reagent and as product.

We can now prove the soundness of the translation of threshold Boolean networks in closed Reaction System. To relate the configurations of a threshold Boolean network with the state of the associated Reaction System we introduce the following definition.

Definition 2. *Given a* threshold Boolean network (M,E) *with* $M = \{S_1, \ldots S_n\}$, *and a state at time t,* $S(t) = \{S_1(t), S_2(t), \ldots S_n(t)\}$. *The translation of the state* $S(t)$ *in a corresponding Reaction System state is given by* $\mathcal{RS}(S(t))$, *defined as follows:* $\mathcal{RS}(S(t)) = \{S_i \mid S_i(t) = 1, i \in [1,n]\}$.

Theorem 1. *Let* (M,E) *be a* threshold Boolean network *with elements* $M = \{S_1, S_2, \ldots S_n\}$ *and threshold parameters* $\Theta = \{\theta_1, \theta_2, \ldots \theta_n\}$. *Given a state at time t,* $S(t) = \{S_1(t), S_2(t), \ldots S_n(t)\}$ *we have that*

$$\mathcal{RS}(S(t+1)) \quad = \quad res_{\mathcal{A}}(\mathcal{RS}(S(t)))$$

where $\mathcal{A} = \mathcal{RS}((M,E)) = (M,A)$ *is the Reaction System obtained by the translation.*

Due to Theorem 1 a threshold Boolean network (M,E) with a state $S(0)$ at time 0 can be simulated by the corresponding closed Reaction System $\mathcal{RS}((M,E))$ by considering the initial state $\mathcal{RS}(S(0))$.

At this point, it important to count the number reactions of the closed Reaction System necessary to simulate a threshold Boolean network (M,E). Such

number depends on the number of the nodes M and of the edges of the network E and also on the threshold parameters Θ. For each $S_i \in M$ the number of the reactions which have S_i as a product depends on the cardinality of $Act(S_i)$ and $In(S_i)$ and on θ_i. Indeed, $Act(S_i)$ and $In(S_i)$ represents the number of the incoming edges which *activates* and *inhibits* S_i respectively.

Proposition 1. *Given a* threshold Boolean network (M, E) *with elements* $M = \{S_1, S_2, \dots S_n\}$ *and threshold parameters* $\Theta = \{\theta_1, \theta_2, \dots \theta_n\}$.
Moreover, let $\mathcal{RS}((M, E)) = (S, A)$ *be the corresponding closed Reaction System.*

- *For each* $i \in \{1, \dots, n\}$, *let* $N(S_i) = \#(\{(R, I, P) \in A \mid S_i \in P\})$ *be the number of the reactions which produce the element* S_i. *We have that*

$$N(S_i) = \begin{cases} \sum_{k=1+\theta_i}^{min(m_i,(l_i+1+\theta_i))} \binom{m_i}{k} \times \binom{l_i}{k-1-\theta_i}, & \text{if } S_i \in M_{nsa}; \\ \sum_{k=1+\theta_i}^{min(m_i,(l_j+1+\theta_i))} \binom{m_i}{k} \times \binom{l_i}{k-1-\theta_i} + & \\ \sum_{h=\theta_i}^{min(m_i,(l_j+\theta_i))} \binom{m_i}{h} \times \binom{l_i}{k-\theta_i}, & \text{if } S_i \in M_{sa}. \end{cases}$$

  where $m_i = \#(Act(S_i))$ *and* $l_i = \#(IN(S_i))$.
- *We have that* $\#(A) = \sum_{i=1}^{n} N(S_i)$.

The previous result is a direct consequence of Definition 1.

Example 3. Let us consider the closed Reaction System $\mathcal{RS}(M, E) = (M, A)$, presented in the Example 2, which is the translation of the threshold Boolean network (M, E) of Example 1. The reaction system has 12 reactions. Indeed, since A, B and C belong to M_{sa} while D belongs to M_{nsa} and the threshold parameter is 0 by applying Proposition 1, we obtain:

$$N(A) = \sum_{i=1}^{min(1,2)} \binom{1}{i} \times \binom{1}{i-1} + \sum_{i=0}^{min(1,1)} \binom{1}{i} \times \binom{1}{i} = 1 + (1 + 1) = 3$$

$$N(B) = \sum_{i=1}^{min(2,2)} \binom{2}{i} \times \binom{1}{i-1} + \sum_{i=0}^{min(2,1)} \binom{2}{i} \times \binom{1}{i} = (2 + 1) + (1 + 2) = 6$$

$$N(C) = \sum_{i=1}^{min(1,1)} \binom{1}{i} \times \binom{0}{i-1} + \sum_{i=0}^{min(1,0)} \binom{1}{i} \times \binom{0}{i} = 1 + 1 = 2$$

$$N(D) = \sum_{i=1}^{min(1,2)} \binom{1}{i} \times \binom{0}{i-1} = 1$$

5 Simulating the Yeast-Cell Cycle Boolean Network

The cell-cycle process by which a cell goes and divides into two cells is a vital process the regulation of which is conserved among the eukaryotes [21]. The process mainly consists in four phases depicted in Figure 2.

In phase G1 the cell grows and, under appropriate conditions, commits to division, in phase S the DNA is synthesized and chromosomes replicated, G2 is the phase where the cell checks the duplicated chromosomes, and finally in the M (Mitosis) phase the cell is divided into two. After the Mitosis phase, the cell enters the G1 phase, hence completing a cycle. There are about 800 genes

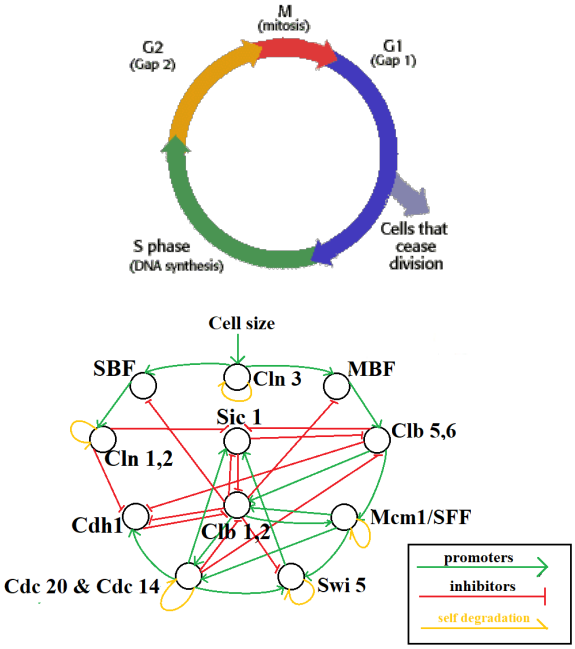


Fig. 3. The Boolean network (M_{Cell}, E_{Cell}).

involved in the cell-cycle process of the budding yeast [25]. However, the number of key regulators that are responsible for the control and regulation of this complex process is much smaller. Based on extensive literature studies, the authors in [20] constructed a network of key regulators involving 11 genes. The relations between genes are described by the boolean network (M_{Cell}, E_{Cell}) depicted in Figure 3, where the threshold parameter θ is always 0. The boolean network was used to study the time evolution of the protein states. Starting from the $2^{11} = 2048$ possible initial states describing a configuration for gene activation, they discover that all of them flow into one of seven attractor stationary states. In particular, among the seven fixed points there is one big attractor that attracts 1764 initial states. We translated the Boolean network (M_{Cell}, E_{Cell}) of Figure 3 into a Reaction System $\mathcal{A}_{cell} = \mathcal{RS}(M_{Cell}, E_{Cell}) = ((M_{Cell}, A_{Cell})$ by applying the procedure described in Definition 1. The Reaction System $\mathcal{A}_{cell}$ has 52 reactions. For simplicity, we just show the translation of reactions describing the production (activation) of a single node of the Boolean network. Consider the central node named `Sic1` in the Boolean network of Figure 3. It has 2 *activating* incoming arcs and 3 *inhibiting* incoming arcs. Since `Sic1` $\in M_{sa}$, by Proposition 1 there will be $\sum_{i=1}^{min(2,4)} \binom{2}{i} \times \binom{3}{i-1} + \sum_{i=0}^{min(2,3)} \binom{2}{i} \times \binom{3}{i} = (2+3)+(1+6+3) = 15$. Indeed, by applying our translation we find the following rules for the production of `Sic1`:

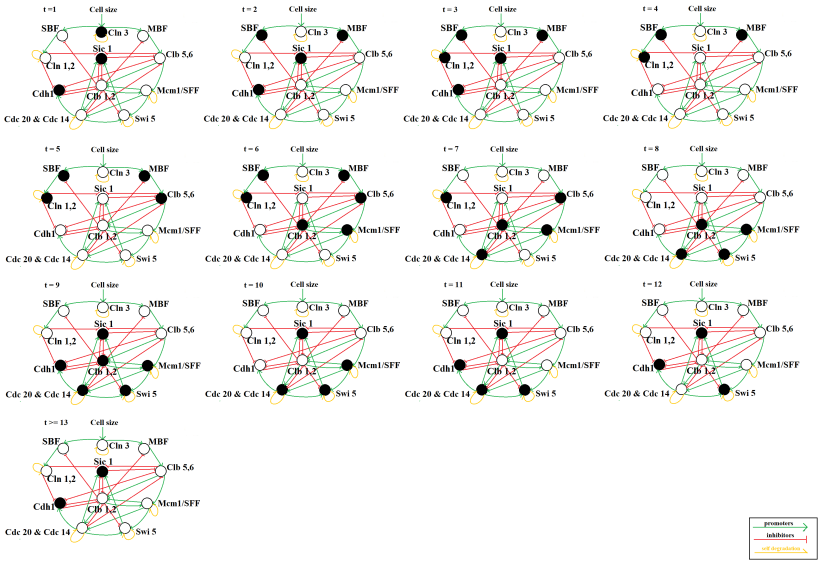


Fig. 4. The cell cycle evolution

$(\{\texttt{Cdc20}\}, \{\texttt{Clb5}, 6, \texttt{Clb1}, 2, \texttt{Cln1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Swi5}\}\{\texttt{Clb5}, 6, \texttt{Clb1}, 2, \texttt{Cln1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Cdc20}, \texttt{Swi5}\}, \{\texttt{Clb1}, 2, \texttt{Cln5}, 6\}, \{\texttt{Sic1}\})$
$(\{\texttt{Cdc20}, \texttt{Swi5}\}, \{\texttt{Clb1}, 2, \texttt{Cln1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Cdc20}, \texttt{Swi5}\}, \{\texttt{Cln1}, 2, \texttt{Clb5}, 6\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}\}, \{\texttt{Clb5}, 6, \texttt{Clb1}, 2, \texttt{Cln1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}, \texttt{Cdc20}\}, \{\texttt{Clb5}, 6, \texttt{Clb1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}, \texttt{Cdc20}\}, \{\texttt{Clb5}, 6, \texttt{Cln1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}, \texttt{Cdc20}\}, \{\texttt{Clb5}, 6, \texttt{Clb1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}, \texttt{Swi5}\}, \{\texttt{Clb5}, 6, \texttt{Clb1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}, \texttt{Swi5}\}, \{\texttt{Clb5}, 6, \texttt{Cln1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}, \texttt{Swi5}\}, \{\texttt{Clb5}, 6, \texttt{Clb1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}, \texttt{Cdc20}, \texttt{Swi5}\}, \{\texttt{Cln1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}, \texttt{Cdc20}, \texttt{Swi5}\}, \{\texttt{Clb1}, 2\}, \{\texttt{Sic1}\})$
$(\{\texttt{Sic1}, \texttt{Cdc20}, \texttt{Swi5}\}, \{\texttt{Clb5}, 6\}, \{\texttt{Sic1}\})$

Note that the behavior of the reactions producing `Sic1` faithfully model the activation of gene `Sic1` in the Boolean network. Consider the case where genes `Cdc20`, `Swi5` and `Clb5, 6` are all active according to the Boolean network of Figure 3 after one step `Sic1` becomes active. The previous state is represented in the Reaction System as the set of activated genes $D_0 = \{\texttt{Cdc20}, \texttt{Swi5}, \texttt{Clb5}, 6\}$. Now starting from D_0, we can apply rule $(\{\texttt{Cdc20}, \texttt{Swi5}\}, \{\texttt{Clb1}, 2, \texttt{Cln1}, 2\}, \{\texttt{Sic1}\})$ to obtain the production(activation) of gene `Sic1`, therefore $\texttt{Sic1} \in \texttt{D}_1$ where $D_0 \rightarrow_{\mathcal{A}_{Cell}} D_1$. Note that if $\texttt{Cln1}, 2$ was also active in the initial state then gene `Sic1` could not be activated according to the Boolean network. This is modeled in the Reaction System by the fact that none of the 15 rules producing `Sic1` could be applied to the set $D_0 = \{\texttt{Cdc20}, \texttt{Swi5}, \texttt{Clb5}, 6, \texttt{Cln1}, 2\}$. Once we have obtained the complete Reaction System $\mathcal{A}_{cell}$ that simulates the entire Boolean network we can run it with any initial state D_0 in order to study the behaviour of the cell when some genes are activated. As a first experiment we executed the Reaction System with the initial state D_0 that it was observed in nature triggers the cell-cycle. Indeed, usually the cell stays in a stationary state where

just genes Sic1 and Cdh1 are active. When the cell grows, the external cell size signal Cell size arrives and activates Cln3. This "excites" the cell from its stationary state and triggers the cycle. We can observe the different states of activation/deactivation of genes during the cell cycle by executing the Reaction System with an initial state where Sic1, Cdh1 and Cln3 are active.
Thus, the following evolution can be observed.

$$\{\texttt{Sic1}, \texttt{Cdh1}, \texttt{Cln3}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{SBF}, \texttt{MBF}, \texttt{Sic1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{SBF}, \texttt{MBF}, \texttt{Sic1}, \texttt{Cln1}, \texttt{2}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{SBF}, \texttt{MBF}, \texttt{Cln1}, \texttt{2}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{SBF}, \texttt{MBF}, \texttt{Cln1}, \texttt{2}, \texttt{Clb5}, \texttt{6}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{SBF}, \texttt{MBF}, \texttt{Cln1}, \texttt{2}, \texttt{Clb5}, \texttt{6}, \texttt{Clb1}, \texttt{2}, \texttt{Mcm1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{Cln1}, \texttt{2}, \texttt{Clb5}, \texttt{6}, \texttt{Clb1}, \texttt{2}, \texttt{Mcm1}, \texttt{Cdc20}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{Clb1}, \texttt{2}, \texttt{Mcm1}, \texttt{Cdc20}, \texttt{Swi5}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{Clb1}, \texttt{2}, \texttt{Mcm1}, \texttt{Cdc20}, \texttt{Swi5}, \texttt{Sic1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{Mcm1}, \texttt{Cdc20}, \texttt{Swi5}, \texttt{Sic1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{Cdc20}, \texttt{Swi5}, \texttt{Sic1}, \texttt{Cdh1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{Swi5}, \texttt{Sic1}, \texttt{Cdh1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{Sic1}, \texttt{Cdh1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{Sic1}, \texttt{Cdh1}\}$$

At this point the evolution reaches the stationary state $\{\texttt{Sic1}, \texttt{Cdh1}\}$ and the cell waits for another external stimulus to arrive, that is an external new cell size signal that activates gene Cln3 and triggers a new cycle. The evolution of the Reaction System represents the evolution of the Boolean network depicted in Figure 4 that describes the entire cell cycle. The Reaction System $\mathcal{A}_{Cell}$ can

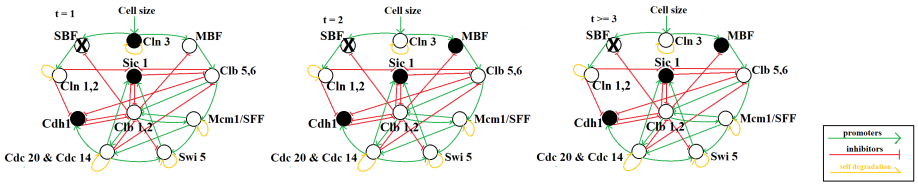


Fig. 5. The cycle evolution where gene SBF was silenced

now be used for studying the influence that each gene has in the cell cycle. Each gene can be silenced in turn simply by deleting the rules that produces such gene. Note that this corresponds to simulate the Boolean network where we canceled the node representing the gene together with all his arcs. As a second example consider the case where gene SBF was silenced. To this aim, let the Reaction System $\mathcal{A}_{Cell} = (M_{Cell}, A_{Cell})$, we consider the Reaction System $\mathcal{A}_{Cell-SBF} = (M_{Cell}, A_{Cell}/\{(R_a, P_a, \{\texttt{SBF}\})|\ a \in A_{Cell}\})$. In this case, starting from the stationary state $\{\texttt{Sic1}, \texttt{Cdh1}, \texttt{Cln3}\}$ the following evolution can be observed.

$$\{\texttt{Sic1}, \texttt{Cdh1}, \texttt{Cln3}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}-\texttt{SBF}}} \{\texttt{Cdh1}, \texttt{MBF}, \texttt{Sic1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}-\texttt{SBF}}} \{\texttt{Cdh1}, \texttt{MBF}, \texttt{Sic1}\}$$

The corresponding evolution on the Boolean network is depicted in Figure 5.

It can be observed that if gene SBF was silenced the cell could not perform the cycle because after one step it reaches a new stationary state. This shows

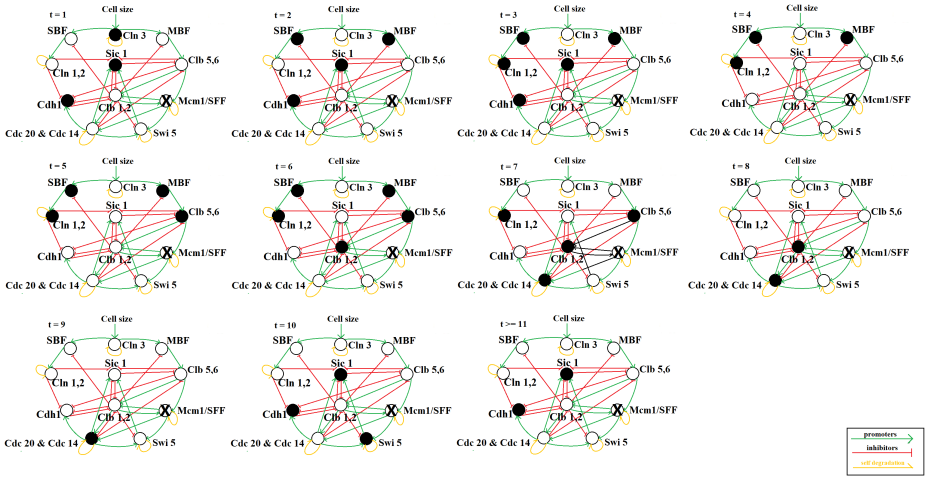


Fig. 6. The cycle evolution where gene `Mcm1` was silenced

that the gene `SBF` was, in some way, necessary for the cycle to be performed. As a last example consider the case where gene `Mcm1` was silenced. The evolution we obtain by considering the Reaction System $\mathcal{A}_{Cell-Mcm1} = (M_{Cell}, A_{Cell}/\{(R_a, P_a, \{\texttt{Mcm1}\})|\ a \in A_{Cell}\})$ starting with the stationary state $\{\texttt{Sic1}, \texttt{Cdh1}, \texttt{Cln3}\}$ is directly depicted in Figure 6.

In this case we obtain a very different result from the previous one. Indeed, it can be observed that even if gene `Mcm1` was silenced the cell could perform most of its cycle and go back to the initial stationary state. This suggests that the gene `Mcm1` was not necessary for the cycle to be performed. Indeed, the cell can recover even if, for some reasons, gene `Mcm1` could not be activated.

6 Conclusions

In this paper we proposed a systematic translation of boolean networks into reaction systems. This allows us to simulate the behaviour of a boolean network simply by executing the reaction system we obtain. We applied our method to model the extensively studied Cell cycle that allows a cell to split. The translation of the boolean network describing the Cell cycle into a reaction system allows us to investigate the role of different genes simply by "playing" with the rules related to the production of such gene. In this way, we studied the effects of the silencing of some genes such `SBF` and `Mcm1` on the entire cell cycle.

However we think that the main advantage of our translation will be the application of well known techniques for detecting dynamic causalities relations in reaction systems (see [14, 2, 1, 3, 4]) to determine causality relations between genes of a boolean network.

References

1. Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo. Specialized predictor for reaction systems with context properties. In *Proc. of the 24th Int. Workshop on Concurrency, Specification and Programming, CS&P 2015*, pages 31–43, 2015.
2. Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo. Investigating dynamic causalities in reaction systems. *Theoretical Computer Science*, 623:114–145, 2016.
3. Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo. Specialized predictor for reaction systems with context properties. *Fundamenta Informaticae*, 147(2-3):173–191, 2016.
4. Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo. Generalized contexts for reaction systems: definition and study of dynamic causalities. *Acta Inf.*, 55(3):227–267, 2018.
5. Roberto Barbuti, Roberta Gori, and Paolo Milazzo. Multiset patterns and their application to dynamic causalities in membrane systems. In *Membrane Computing - 18th International Conference, CMC 2017, Bradford, UK, July 25-28, 2017, Revised Selected Papers*, pages 54–73, 2017.
6. Roberto Barbuti, Roberta Gori, and Paolo Milazzo. Predictors for flat membrane systems. *Theor. Comput. Sci.*, 736:79–102, 2018.
7. A. Bernasconi, V. Ciriani, R. Drechsler, and T. Villa. Logic Minimization and Testability of 2-SPP Networks. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(7):1190–1202, 2008.
8. A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli. Synthesis of autosymmetric functions in a new three-level form. *Theory of Computing Systems*, 42(4):450–464, 2008.
9. A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa. *Logic synthesis by signal-driven decomposition*. 2011.
10. Anna Bernasconi, Valentina Ciriani, Gabriella Trucco, and Tiziano Villa. Using Flexibility in P-Circuits by Boolean Relations. *IEEE Trans. Computers*, 64(12):3605–3618, 2015.
11. Chiara Bodei, Roberta Gori, and Francesca Levi. An analysis for causal properties of membrane interactions. *Electr. Notes Theor. Comput. Sci.*, 299:15–31, 2013.
12. Chiara Bodei, Roberta Gori, and Francesca Levi. Causal static analysis for brane calculi. *Theor. Comput. Sci.*, 587:73–103, 2015.
13. Robert Brijder, Andrzej Ehrenfeucht, Michael G. Main, and Grzegorz Rozenberg. A tour of reaction systems. *Int. J. Found. Comput. Sci.*, 22(7):1499–1517, 2011.
14. Robert Brijder, Andrzej Ehrenfeucht, and Grzegorz Rozenberg. A note on causalities in reaction systems. *ECEASST*, 30, 2010.
15. Nadia Busi. Causality in membrane systems. In *Membrane Computing, 8th International Workshop, WMC 2007, Thessaloniki, Greece, June 25-28, 2007 Revised Selected and Invited Papers*, pages 160–171, 2007.
16. Bernard Derrida. Dynamical phase transition in nonsymmetric spin glasses. *Journal of Physics A: Mathematical and General*, 20(11):L721–L725, 1987.
17. Andrzej Ehrenfeucht and Grzegorz Rozenberg. Reaction systems. *Fundamenta informaticae*, 75(1-4):263–280, 2007.
18. Roberta Gori and Francesca Levi. Abstract interpretation based verification of temporal properties for bioambients. *Inf. Comput.*, 208(8):869–921, 2010.
19. K.E. Kürten. Critical phenomena in model neural networks. *Physics Letters A*, 129(3):157 – 160, 1988.

20. Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences*, 101(14):4781–4786, 2004.
21. Andrew Murray and Tim Hunt. *The Cell Cycle*. Oxford Univ.Press, New York, 1993.
22. Thimo Rohlf and Stefan Bornholdt. Criticality in random threshold networks: annealed approximation and beyond. *Physica A: Statistical Mechanics and its Applications*, 310(1):245 – 259, 2002.
23. R. Rudell and A. Sangiovanni-Vincentelli. Multiple Valued Minimization for PLA Optimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 6(5):727–750, 1987.
24. Thomas Schlitt and Alvis Brazma. Current approaches to gene regulatory network modelling. *BMC bioinformatics*, 8(6):S9, 2007.
25. P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O Brown., D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9(12):3273–3297, 1998.

Hidden States in Reaction Systems ⋆

Roberta Gori[1], Damas Gruska[2], and Paolo Milazzo[1]

[1] Dipartimento di Informatica, Università di Pisa, Italy
[2] Department of Applied Informatics, Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, Slovak Republic

Abstract. Pushing forward a previous investigation on security of reaction systems, we introduce new state based security properties. Assume there are some states of a reaction system that are in some sense critical, and that we want to hide whether the system reaches them. We define new security properties that guarantee that an external observer who has only a partial knowledge on the objects provided by the environment cannot infer whether a secret state is reached by the system. We also propose an effective method for verifying such properties. The verification method is based on a newly defined extension of the concept of formula based predictor to set of states.

1 Introduction

Reaction systems is a *qualitative* modeling formalism introduced by Ehrenfeucht and Rozenberg to model biological systems [1, 2]. It is based on the two opposite mechanisms of *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. A *reaction system* is essentially a set of rewrite rules (*reactions*) having the form (R, I, P), where R, I and P are sets of objects representing reactants, inhibitors and products, respectively. The state of a reaction system is a finite set of objects, describing the biological entities that are present in the modeled system. The presence of an object in the state means that the corresponding biological entity is present in a number of copies as high as needed. This is the *threshold supply* assumption and characterizes reaction systems.

A reaction system evolves by means of the application of its reactions.The threshold supply assumption ensures that all the applicable reactions in a step are always applied, since they do not compete for thier reactants. The application of a set of reactions results in the introduction of all of their products in the next state of the system. The behaviour of a reaction system is driven by the (set of) contextual elements which are provided by the external environment at each step. Such elements join the current state of the system and can enable or disable reactions. The computation of the next state of a reaction system is a deterministic procedure. Consequently, if the contextual elements provided to the system at each step are know, then the whole execution of the system is determined. On the other hand, if they are not known, the overall system dynamics becomes non deterministic.

⋆ Work supported by the grant VEGA 1/0778/18 and by the project "Metodologie informatiche avanzate per l'analisi di dati biomedici" (University of Pisa, PRA 2017_44).

In previous papers [3, 4] we investigated the concept of opacity in reaction systems. Assume we have a real biochemical system described by a reaction system, and an observer having a partial information about the objects provided by the environment because of the cost of obtaining such informaton. We can distinguish two types of objects: visible *low level* (L) objects, and invisible *high level* (H) objects. We studied the detectability of H-objects, namely how much information on the presence of H-objects can be obtained by just observing the presence of L-objects in context sequences. This problem, called information flow [5], was extensively studied in security by introducing the concept of *opacity* [6, 7]. We reformulated opacity for reaction systems and proposed dynamic causality relatioships (formula based predictors) as an effective method to verify opacity properties in reaction systems.

In this paper we push forward our approach by considering sets of *secret states*. Let Sec be a set of states and assume we want to hide to an external observer whether a reaction system reaches one of such states. So, Sec is a set of secret states. As before, the observer can only see L-objects in context sequences. In order to prevent the observer to infer whether the system reaches a secret state, we have to ensure that for every context sequence leading to one of such states there exists another context sequence leading to a non-secret state that is indistinguishable from the previous one from the (limited) point of view of the observer. In other words, the two context sequences must make the same use of L-objects, which are the only ones that can be observed. We will formalize this idea in terms of two security properties called Current State Opacity and n-p Window Current state Opacity, and we will provide effective methods for verifying them based on dynamic causalities.

Dynamic causalities deal with the ways entities dynamically influence each other. Brijder, Ehrenfeucht and Rozenberg initiated an investigation on *causalities* in reaction systems [8], by introducing the idea of *predictor*. Assume that one is interested in knowing whether a particular object $s \in S$ will be present after n steps of execution of the reaction system. Since the only source of non-determinism are the contextual elements received at each step, knowing which objects will be received allows the production of s after n steps to be predicted. In [9–12] the new notion of *formula based predictor* was introduced. A formula based predictor is 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.* In the style of [13, 14], here the notion of formula-based predictor is first naturally extended to sets of objects (states), and then it is extended to sets of states. The result is a formula based predictor that can be used to precisely characterize all context sequences that lead to one secret state in a set Sec. We apply this extended predictor for secret states in Sec to prove whether the reaction system is opaque for an observer that can only see L-objects of the context sequence provided by the environment.

2 Reaction Systems

In this section we briefly introduce reaction systems [1, 2]. Given S, a finite set of symbols, called objects, a *reaction* is a triple (R, I, P) with $R, I, P \subseteq S$, composed

of *reactants* R, *inhibitors* I, and *products* P. Reactants and inhibitors are disjoint ($R \cap I = \emptyset$) otherwise the reaction would never be applicable. The set of all possible reactions over a set S is denoted by $\mathrm{rac}(S)$. Finally, a *reaction system* is a pair $\mathcal{A} = (S, A)$, with S a finite background set, and $A \subseteq \mathrm{rac}(S)$ a set of reactions.

The state of a reaction system is a set of objects. Let $a = (R_a, I_a, P_a)$ be a reaction and T be a set of objects. The result $\mathrm{res}_a(T)$ 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 for which no inhibitor is present in the current state is applied, and the result of application of multiple reactions is cumulative. Given a reaction system $\mathcal{A} = (S, A)$, the result of the application of A to a set $T \subseteq S$ is defined as $\mathrm{res}_{\mathcal{A}}(T) = \mathrm{res}_A(T) = \bigcup_{a \in A} \mathrm{res}_a(T)$. An important characteristic of reaction systems is the assumption about the *non-permanency* of objects: the objects carried over to the next step are only those produced by reactions. All the other objects vanish.

The dynamics of a reaction system $\mathcal{A} = (S, A)$ is driven by the *contextual* objects, namely the objects which are supplied to the system by the external environment at each step. The dynamics is defined as an *interactive process* $\pi = (\gamma, \delta)$, with γ and δ 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 $D_i = res_{\mathcal{A}}(C_{i-1} \cup D_{i-1})$ for all $1 \leq i \leq n$. Finally, the *state sequence* of π is 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 call $\gamma = C_0, C_1, \ldots, C_n$ a n-step context sequence.

3 Preliminaries on Predicate Logic

The aim of formula based predictors is to characterize all context sequences that lead to the production of a specific object in a given number of steps. In order to describe conditions on the presence or absence of objects in context sequences, we use objects of reaction systems as propositional symbols. Formally, we define the set F_S of propositional formulas on S in the standard way: $S \cup \{true, false\} \subseteq F_S$ and $\neg f_1, f_1 \vee f_2, f_1 \wedge f_2 \in F_S$ if $f_1, f_2 \in F_S$. Propositional formulas F_S are interpreted with respect to subsets of S. Intuitively, a subset $C \subseteq S$ is used to describe the objects that are present in (an element of) a context sequence, and this implies the truth of the corresponding propositional symbol. The formal definition of the satisfaction relation 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* $s \in C$, $\quad C \models true$,
$C \models \neg f'$ *iff* $C \not\models f'$, $\quad C \models f_1 \vee f_2$ *iff either* $C \models f_1$ *or* $C \models f_2$,
$C \models f_1 \wedge f_2$ *iff* $C \models f_1$ *and* $C \models f_2$.

In the following $\equiv_l$ stands for the logical equivalence on propositional formulas F_S. Moreover, given a formula $f \in F_S$ we use $atom(f)$ to denote the set of propositional symbols that appear in f. The simplified version of a formula is obtained by applying the standard formula simplification procedure of propositional logic converting a formula to Disjunctive Normal Form, $\mathcal{DNF}(f)$. We recall that for any formula $f \in F_S$ the simplified formula $\mathcal{DNF}(f)$ is equivalent to f, it is minimal with respect to the number of propositional symbols and unique up to commutativity and associativity. Thus, we have $f \equiv_l \mathcal{DNF}(f)$ and $atom(\mathcal{DNF}(f)) \subseteq atom(f)$ and there exists no formula f' such that $f' \equiv_l f$ and $atom(f') \subset atom(\mathcal{DNF}(f))$.

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 formula 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 $ap(a)$, is defined as follows: $ap(a) = \left(\bigwedge_{s_r \in R} s_r\right) \wedge \left(\bigwedge_{s_i \in I} \neg s_i\right)$.*

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

We introduce a simple reaction system as running example.

Example 1. Let $\mathcal{A}_1 = (\{A, \ldots, G\}, \{a_1, a_2, a_3\})$ be a reaction system with

$$a_1 = (\{A\}, \{\}, \{B\}) \qquad a_2 = (\{C, D\}, \{\}, \{E, F\}) \qquad a_3 = (\{G\}, \{B\}, \{E\}) \ .$$

The *applicability predicates* of the reactions are $ap(a_1) = A$, $ap(a_2) = C \wedge D$ and $ap(a_3) = G \wedge \neg B$. Thus, the *causal predicates* of the objects are

$$cause(A) = cause(C) = cause(D) = cause(G) = false,$$
$$cause(B) = A,\ cause(F) = C \wedge D,\ cause(E) = (G \wedge \neg B) \vee (C \wedge D).$$

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

4 Formula Based Predictors

In the first part of this section we introduce the notion of *formula based predictor* as it was originally presented in [9]. Then, we extend the notion of predictors to

[3] We assume that $cause(s) = false$ if there is no $(R, I, P) \in A$ such that $s \in P$.

states (see Corollary 1) and to sets of states (see Corollary 2) in order to address causal dependences of the secret states set Sec that we want to hide.

A formula based predictor for an object s at step $n+1$ is a propositional formula satisfied 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 $S^n = \bigcup_{i=0}^{n} S_i$ 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. Similarly, the set F_{S_i} can be defined by replacing S with S_i. Given a formula $f \in F_S$, a corresponding formula $\mathit{labelled}(f,i) \in F_{S_i}$ can be obtained by replacing each $s \in S$ in f with $s_i \in S_i$.

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 γ satisfies the formula $A_0 \wedge B_1$ (i.e. $\gamma \models A_0 \wedge B_1$) while γ does not satisfy the formula $A_0 \wedge (\neg B_1 \vee C_1)$ (i.e. $\gamma \not\models A_0 \wedge (\neg B_1 \vee C_1)$).

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

Definition 5 (Formula based Predictor). *Let $\mathcal{A} = (S, A)$ be a reaction system, $s \in S$ 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 $\gamma = C_0, \ldots, C_n$*

$$\gamma \models f \Leftrightarrow s \in D_{n+1}$$

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

Note that if formula f f-predicts s in $n+1$ steps and if $f' \equiv_l f$ 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 $f_1 \equiv_l f_2$ and $atom(f_1) \subseteq atom(f_2)$.*

In [9] it is shown that there exists a *unique equivalence class* of formula based predictors for s in $n+1$ steps that is minimal with respect to the order $\sqsubseteq_f$.

We now define an operator `fbp` that allows formula based predictors to be effectively computed.

Definition 7. *Let $\mathcal{A} = (S, A)$ be a r.s. and $s \in S$. We define a function* $\mathtt{fbp} : S \times \mathbb{N} \to F_{S^n}$ *as follows:* $\mathtt{fbp}(s, n) = \mathtt{fbs}(cause(s), n)$*, where the auxiliary function* $\mathtt{fbs} : F_S \times \mathbb{N} \to F_{S^n}$ *is recursively defined as follows:*

$$\begin{array}{ll}
\mathtt{fbs}(s, 0) = s_0 & \mathtt{fbs}(s, i) = s_i \vee \mathtt{fbs}(cause(s), i-1) \quad \text{if } i > 0 \\
\mathtt{fbs}((f'), i) = (\mathtt{fbs}(f', i)) & \mathtt{fbs}(f_1 \vee f_2, i) = \mathtt{fbs}(f_1, i) \vee \mathtt{fbs}(f_2, i) \\
\mathtt{fbs}(\neg f', i) = \neg \mathtt{fbs}(f', i) & \mathtt{fbs}(f_1 \wedge f_2, i) = \mathtt{fbs}(f_1, i) \wedge \mathtt{fbs}(f_2, i) \\
\mathtt{fbs}(true, i) = true & \mathtt{fbs}(false, i) = false
\end{array}$$

The function fbp gives a formula based predictor that, in general, may not be minimal with respect to the approximation order $\sqsubseteq_f$. Therefore, the calculation of a minimal formula based predictor requires the application of the standard simplification procedure that simplifies the obtained logic formula and puts it in disjunctive normal form, here called simply $\mathcal{DNF}(.)$.

Theorem 1. *Let $\mathcal{A} = (S, A)$ be a r.s.. For any object $s \in S$,*

- $\mathtt{fbp}(s, n)$ f-predicts s *in* $n + 1$ *steps;*
- $\mathcal{DNF}(\mathtt{fbp}(s, n))$ f-predicts s *in* $n + 1$ *steps and is* minimal *w.r.t.* $\sqsubseteq_f$.

Example 2. Let us consider again the reaction system $\mathcal{A}_1$ 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 fbp:

$$\begin{array}{rl}
\mathtt{fbp}(E, 3) = & \mathtt{fbs}\big((G \wedge \neg B) \vee (C \wedge D), 3\big) \\
= & \big(\mathtt{fbs}(G, 3) \wedge \neg \mathtt{fbs}(B, 3)\big) \vee \big(\mathtt{fbs}(C, 3) \wedge \mathtt{fbs}(D, 3)\big) \\
= & \big((G_3) \wedge \neg(B_3 \vee \mathtt{fbs}(A, 2))\big) \vee (C_3 \wedge D_3) \\
= & \big(G_3 \wedge \neg B_3 \wedge \neg A_2\big) \vee (C_3 \wedge D_3)\big)
\end{array}$$

A context sequence satisfies $\mathtt{fbp}(E, 3)$ 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 w.r.t. $\sqsubseteq_f$. This is because $\mathcal{DNF}(\mathtt{fbp}(E, 3)) = \mathtt{fbp}(E, 3)$. Indeed, the formula $\mathtt{fbp}(E, 3)$ cannot be further simplified and any literal cannot be canceled without obtaining a non equivalent formula.

The result of Theorem 1 can be easily extended to states. Indeed, we can characterize all the context sequences that lead to the production of the set of objects of the state. To this aim, we need to consider the context sequences that satisfy *all conditions for the production of each single object* of the set. Assume *sec* to be a state, that is, a set of objects in S then we can characterize all the context sequence leading to states in the following way.

Corollary 1. *Let $\mathcal{A} = (S, A)$ be a r.s.. Consider sec a set of objects in S,*

- $\bigwedge_{s \in sec} \mathtt{fbp}(s, n)$ f-predicts *sec in* $n + 1$ *steps;*
- $\mathcal{DNF}\left(\bigwedge_{s \in sec} \mathtt{fbp}(s, n)\right)$ f-predicts s *in* $n + 1$ *steps and is* minimal *w.r.t.* $\sqsubseteq_f$.

Moreover, the previous results can be extended to finite sets of states. Assume Sec to be a set of states $\{sec_1, sec_2, ..., sec_m\}$, for some m. We need to characterize all the context sequences that lead to some state in Sec.

Corollary 2. *Let $\mathcal{A} = (S, A)$ be a r.s.. Let Sec be a set of states $\{sec_1, sec_2, ..., sec_m\}$, for some m,*

- $\bigvee_{sec_i \in Sec} \left(\bigwedge_{s \in sec_i} \texttt{fbp}(s, n)\right)$ f-predicts *the set Sec in $n+1$ steps;*
- $\mathcal{DNF}\left(\bigvee_{sec_i \in Sec} \left(\bigwedge_{s \in sec_i} \texttt{fbp}(s, n)\right)\right)$ f-predicts *Sec in $n+1$ steps and is* minimal *w.r.t.* $\sqsubseteq_f$.

Example 3. Let us consider again the reaction system $\mathcal{A}_1$ of Examples 1 and 2. Assume we are interested in reaching the state $\{E, F\}$ after 4 steps. In Example 2 we calculated the logic formula that *f-predicts E in 4 steps* applying the function `fbp`. This resulted in the formula $\big(G_3 \wedge \neg B_3 \wedge \neg A_2\big) \vee (C_3 \wedge D_3)\big)$. Analogously, we can calculate the logic formula that *f-predicts F in 4 steps* applying the function `fbp`. This resulted in the formula $(C_3 \wedge D_3)$. Now, in order to obtain the minimal formula characterising the context sequences that lead to the state where both E and F are present, according to Corollary 1, we need to compute

$$\mathcal{DNF}(\texttt{fbp}(E, 4) \wedge \texttt{fbp}(F, 4)) =$$
$$\mathcal{DNF}\Big(\big((G_3 \wedge \neg B_3 \wedge \neg A_2) \vee (C_3 \wedge D_3)\big) \wedge (C_3 \wedge D_3)\Big) = C_3 \wedge D_3.$$

A context sequence satisfies $C_3 \wedge D_3$ iff the execution of the reaction system leads to the production of both object E and F after 4 steps.

Assume now we are interested in characterising the context sequences that either lead to state $\{E, F\}$ or to state $\{B\}$ after 4 steps. Hence, in this case $Sec = \{\{E, F\}, \{B\}\}$.

According to Corollary 2 the minimal formula can be obtained by computing

$$\mathcal{DNF}\left((\texttt{fbp}(E, 4) \wedge \texttt{fbp}(F, 4)) \vee \texttt{fbp}(B, 4)\right) =$$
$$\mathcal{DNF}((C_3 \wedge D_3) \vee A_3) = (C_3 \wedge D_3) \vee A_3.$$

Note that any sequence satisfying the formula $(C_3 \wedge D_3) \vee A_3$ leads to a state in *Sec*. Moreover, such sequences are the only ones that can lead to the production of a state in *Sec*.

5 Information flow

As in [3, 4], we now consider a reaction system $\mathcal{A} = (\mathcal{S}, \mathcal{A})$ where we assume an external observer can only detect or see some kinds of objects in the context sequence. To formally describe this situation, borrowing techniques developed for reasoning about flow based security (see [5]), we divide objects from S into two groups, namely public (low level) objects L and private (high level) objects H. It is assumed that $L \cup H = S$ and $L \cap H = \emptyset$. We assume that an observer can see only L-objects, i.e. objects from L. Moreover, we introduce an equivalence on sets of objects and on contexts sequences. Two sets of objects A, B are equivalent with respect to the set M if they contain the same objects apart from those in M. Formally, $A \equiv_M B$ iff $A \setminus M = B \setminus M$. This can be applied to reaction system contexts: we write $\gamma_1 \equiv_M \gamma_2$ if $\gamma_1 = C_0^1, ...C_n^1, ...$ and $\gamma_2 = C_0^2, ...C_n^2, ...$ and $\forall i, C_i^1 \equiv_M C_i^2$. To formalize information flow between L-objects and H-objects we exploit a concept known as *current state opacity* (see [15] for an overview paper).

5.1 Current State Opacity

Let us assume a set of states Sec with $Sec \subset 2^S$. We assume an external observer of the system who can detect or see only L objects in the context sequence, but who wants to discover whether the current state W_i is a secret state belonging to Sec. In this context a reaction system is i-current state opaque if whenever there exists a context sequence leading to a secret state of Sec, there exists an equivalent (with respect to the L object) context sequence that does not lead to a secret state in Sec. This will assure us that just observing the context sequence an external observer cannot decide whether the system will go to a secret state.

Definition 8. *(i-Step Current State Opacity) The reaction system $\mathcal{A} = (S, A)$ is i-current state opaque with respect to L and Sec iff whenever there exists an i-step context sequence γ leading to a secret state in Sec, that is, $D_{i+1} \in Sec$, there also exists an i-step context sequence γ' not leading to a state in Sec, that is, $D'_{i+1} \notin Sec$, such that $\gamma \equiv_L \gamma'$.*

Note that differently from our previous work [3, 4], here the attacker observes properties of context sequences to detect properties of the system states.

Since formula based predictors express all causal dependences of an object from all the objects of the context sequences, we can use this concept to verify if a reaction system is i-step current state opaque.

Theorem 2. *A r.s. $\mathcal{A}$ is i-current state opaque with respect to L and Sec iff*

$$\mathcal{DNF}\Big(\bigvee_{sec \in Sec}\big(\bigwedge_{s \in sec} \mathtt{fbp}(s,i)\big)\Big) = c_1 \vee c_2 \vee ... \vee c_n \text{ and}$$
$$\forall m \in \{1,...,n\}, \{A \mid A_j \in atom(c_m), \text{ with } 0 \leq j < i\} \cap (S \setminus L) \neq \emptyset.$$

Proof. We start by proving the right hand implication. Assume by contradiction that the reaction system $\mathcal{A}$ is i-current state opaque with respect to L and Sec but that there exists a c_m such that $\{A \mid A_j \in atom(c_m), \text{ with } 0 \leq j < i\} \cap (S \setminus L) = \emptyset$. Choose a minimal context sequence γ such that $\gamma \models c_m$. γ has to be minimal in the sense that it just provides the positive literals in the conjunction c_m. Note that by hypothesis, γ provides only low level L-objects. Note that $\gamma \models c_m$ implies that $\gamma \models c_1 \vee c_2 \vee ... \vee c_n = \mathcal{DNF}\Big(\bigvee_{sec \in Sec}\big(\bigwedge_{s \in sec} \mathtt{fbp}(s,i)\big)\Big)$. By applying Corollary 2 we have that the context sequence γ leads to the production of one state in Sec after i steps. However, since γ contains just low level objects L, any context sequence γ' such that $\gamma' \equiv_L \gamma$ will satisfy c_m, since c_m contains only L-objects. Then, by Corollary 2 any γ' will lead to the production of a state in Sec after i steps. Therefore $\mathcal{A}$ is not i-current state opaque. This gives a contradiction.

For proving the left hand implication, assume, by contradiction that every c_i contain at least an H object but that the reaction system $\mathcal{A}$ is not i-current state opaque with respect to a set of low level objects L and Sec. This implies that there do not exist two context sequence γ and γ' with $\gamma \equiv_L \gamma'$ such that one lead to a secret state in Sec and the other does not.

Choose a γ leading to the production of a state in Sec such that it satisfy only one particular conjunction $c_{\bar{i}}$ in the disjunction $c_1 \vee c_2 \vee ... \vee c_n$. By Corollary 2 such

γ exists and we can choose γ as the minimal context sequence satisfying a $c_{\bar{i}}$. Since by hypothesis $c_{\bar{i}}$ is a conjunction containing at least one object in $S \setminus L$ consider γ' as the context sequence satisfying the conjunction of low level objects in $c_{\bar{i}}$ but that does not satisfy the $S \setminus L$ literals in $c_{\bar{i}}$. Now, by construction $\gamma \equiv_L \gamma'$. However, $\gamma' \not\models c_{\bar{i}}$. Moreover, since we have chosen γ to be the minimal context sequence satisfying just $c_{\bar{i}}$ and $c_{\bar{i}} \in c_1 \vee c_2 \vee ... \vee c_n$ then it is simplified, we can be sure that $\gamma' \not\models c_1 \vee c_2 \vee ... \vee c_n$. Then, by Corollary 2, we have that the context sequence γ' does not lead to the production of a state in *Sec*. Hence, we found γ and γ' such that $\gamma \equiv_L \gamma'$ and context sequence γ leads to a secret state in *Sec* while context sequence γ' does not. This gives a contradiction. □

This gives us an easy method to verify if a reaction system is i-current state opaque with respect to a set of low level objects L and a secret set of states *Sec*. While computing $c_1 \vee c_2 \vee ... \vee c_n$ gives us a way to represent all different context sequences that lead to the production of a secret state in *Sec* (see Corollary 2), the condition that each conjunction in $c_1 \vee c_2 \vee ... \vee c_n$ has to contain at least one non low level object, gives us a way to automatically construct an L-equivalent context sequence that does not lead to a state in *Sec*. We will illustrate this construction in the next example. As a consequence of Theorem 2, we can state the following proposition.

Proposition 1. *The property of a reaction system $\mathcal{A}$ to be i-current state opaque with respect to a set of low level objects L and a secret set of states Sec is decidable.*

Example 4. Let $\mathcal{A}_2 = (\{A, \ldots, F\}, \{a_1, a_2, a_3, a_4\})$ be a reaction system with

$$a_1 = (\{A\}, \{B\}, \{C\}) \qquad a_2 = (\{A\}, \{D\}, \{C\})$$
$$a_3 = (\{D\}, \{\}, \{B\}) \qquad a_4 = (\{F\}, \{\}, \{E\})$$

and consider $L = \{A, B, E, F\}, Sec = \{\{C, E\}\}$. Note that $\mathcal{A}_2$ is 3-current state opaque even if E is caused just by a low level object F. Roughly speaking, $\mathcal{A}_2$ is i-current state opaque for each $i \geq 2$ because in that case C is always caused by an H level object. This can formally be proved by considering $\mathcal{DNF}(\texttt{fbp}(C,3) \wedge \texttt{fbp}(E,3))$

$$\begin{aligned}
\mathcal{DNF}(\texttt{fbp}(C,3) \wedge \texttt{fbp}(E,3)) &= \mathcal{DNF}\big(\texttt{fbs}\big((A \wedge \neg B) \vee (A \wedge \neg D), 3\big) \wedge \texttt{fbs}(F,3)\big) \\
&= \mathcal{DNF}\big(((\texttt{fbs}(A,3) \wedge \neg \texttt{fbs}(B,3)) \\
&\qquad \vee (\texttt{fbs}(A,3) \wedge \neg \texttt{fbs}(D,3))) \wedge \texttt{fbs}(F,3)\big) \\
&= \mathcal{DNF}\big(((A_3 \wedge \neg B_3 \wedge D_2) \vee (A_3 \wedge \neg D_3)) \wedge F_3\big) \\
&= (A_3 \wedge \neg B_3 \wedge D_2 \wedge F_3) \vee (A_3 \wedge \neg D_3 \wedge F_3)
\end{aligned}$$

Since both conjunctions $A_3 \wedge \neg B_3 \wedge D_2 \wedge F_3$ and $A_3 \wedge \neg D_3 \wedge F_3$ contain at least a high level object D then by Theorem 2 we are sure that $\mathcal{A}_2$ is 3-current state opaque.

It is worth noting that using the formula based predictor for each γ leading to the production of a secret state in *Sec* we can actually construct γ' with $\gamma \equiv_L \gamma'$ such that γ' does not lead to a secret state in *Sec*. Indeed, let $\gamma = C_1, C_2, C_3$ where C_2 and C_3 are such that $D \in C_2$, $F, A \in C_3$ and $B \notin C_3$. Consider then $\gamma' = C_1, C_2 \setminus \{D\}, C_3$, by Corollary 2, we have that γ lead to a state in *Sec* while γ' does not lead to the state in *Sec*.

The following example shows that the conditions for a system to be i-current state opaque cannot be checked on isolation. Let $\mathcal{A}_3 = (\{A, \ldots, D\}, \{a_1, a_2\})$ be the following reaction system with rules

$$a_1 = (\{A\}, \{D\}, \{B\}) \qquad a_2 = (\{A, D\}, \{\}, \{C\})$$

and consider $L = \{A, B, C\}, Sec = \{\{C\}\{B\}\}$. Note that both rules depend on one H-object D. However, the system is not i-current state opaque for any $i \geq 1$. Let us verify if a system is 3-current state opaque,

$$\begin{aligned}\mathcal{DNF}(\mathtt{fbp}(B,3) \vee \mathtt{fbp}(C,3)) &= \mathcal{DNF}\big(\mathtt{fbs}\big((A \wedge \neg D), 3\big) \vee \mathtt{fbs}\big((A \wedge D), 3\big)\big) \\ &= \mathcal{DNF}\big((\mathtt{fbs}(A,3) \wedge \neg \mathtt{fbs}(D,3) \\ &\qquad \vee (\mathtt{fbs}(A,3) \wedge \mathtt{fbs}(D,3))\big) \\ &= \mathcal{DNF}\big((A_3 \wedge \neg D_3) \vee (A_3 \wedge D_3)\big) = A_3\end{aligned}$$

In this case, the conjunction A_3 does not satisfy the claim of Theorem 2 since it does not have at least one hight level H-object. Indeed, consider any context sequence $\gamma = C_1, C_2, C_3$ where $A \in C_3$. Note that any context sequence $\gamma' \equiv_L \gamma$ will provide A at the third step. Then, by Corollary 2, any $\gamma' \equiv_L \gamma$ will lead to the state in Sec. Hence, $\mathcal{A}_3$ is not 3-state opaque.

5.2 *n-p* Window State Opacity

We now introduce a stronger notion of opacity. Assume now that an observer can observe all objects in the context sequence except for a "blurry window" on which it can observe just L-objects. Once again he wants to discover whether the state at some given step belongs to the set of secret states Sec.

We first define the concept of observational window of a context sequence. Let $\gamma = C_0, ...C_n, ..., C_p, ..., C_i$, by $\gamma_{n,p}$, for $0 \leq n \leq p$ we denote the subsequence $C_n, ..., C_p$.

Definition 9. *(n-p Window i-State Opacity) Let n and p such that $0 \leq n \leq p \leq i$.*

Reaction system $\mathcal{A} = (S, A)$ is n-p window i-state opaque with respect to L and Sec, iff whenever there exists a γ such that D_{i+1} belongs to Sec, i.e. $D_{i+1} \in Sec$, there exists γ' such that state D'_{i+1} does not belong to Sec i.e. $D_{i'+1} \notin Sec$ and $\gamma_{0,n-1} \equiv_S \gamma'_{0,n-1}$, $\gamma_{n,p} \equiv_L \gamma'_{n,p}$ and $\gamma_{p+1,i} \equiv_S \gamma'_{p+1,i}$.

Once again, formula based predictors can be used to verify if a reaction system is n-p window i-state opaque.

Theorem 3. *A reaction system $\mathcal{A}$ is n-p window i-state opaque with respect to L and Sec iff for every*

$$\begin{aligned}&\mathcal{DNF}\Big(\textstyle\bigvee_{sec \in Sec}\big(\bigwedge_{s \in sec} \mathtt{fbp}(s,i)\big)\Big) = c_1 \vee c_2 \vee ... \vee c_n \text{ and} \\ &\forall m \in \{1, ..., n\}, \{A \mid A_j \in atom(c_m), \text{ with } n \leq j \leq p\} \cap (S \setminus L) \neq \emptyset.\end{aligned}$$

As before, to verify if a reaction system is n-p window i-state opaque with respect to a set of low level objects L and a secret set of states Sec, we can check $c_1 \vee c_2 \vee ... \vee c_n$.

Proof. The proof is similar to the proof of Theorem 2, therefore it is only sketched.

For the right hand implication assume by contradiction that the reaction system $\mathcal{A}$ is n-p window i-state opaque with respect to L and Sec but the second part of the claim is false for at lest one c_m. Choose a minimal (in the sense of the proof of Theorem 2) context sequence γ such that $\gamma \models c_m$. By hypothesis, γ does not provide $S \setminus L$ objects at any step included between n and p. Note that any context sequence γ' such that $\gamma_{0,n-1} \equiv_S \gamma'_{0,n-1}$, $\gamma_{n,p} \equiv_L \gamma'_{n,p}$ and $\gamma_{p+1,i} \equiv_S \gamma'_{p+1,i}$ will satisfy c_m. Then, by Corollary 2 any γ' will lead to the production of a state in Sec after i steps. This gives a contradiction.

For proving the left hand implication, assume, by contradiction that every c_i contain at least one $S \setminus L$ object at some step included between n and p but $\mathcal{A}$ is not n-p window i-state opaque. This means that there do not exist two context sequence γ and γ' with $\gamma_{0,n-1} \equiv_S \gamma'_{0,n-1}$, $\gamma_{n,p} \equiv_L \gamma'_{n,p}$ and $\gamma_{p+1,i} \equiv_S \gamma'_{p+1,i}$ such that one lead to a secret state in Sec and the other does not.

Choose a $\gamma = C_0, ..., C_i$ leading to the production of a state in Sec such that it satisfies only one particular conjunction $c_{\bar{i}}$ in the disjunction $c_1 \vee c_2 \vee ... \vee c_n$. By Corollary 2 such γ exists. Consider $\gamma' = C_0, ..., C_{n-1}, C'_n, ..., C'_p, C_{p+1}, ..., C_i$ as the context sequence such that $C'_n, ..., C'_p$, satisfy the conjunction of low level objects only included between n and p of $c_{\bar{i}}$ but that does not satisfy the $S \setminus L$ literals of $c_{\bar{i}}$. Now, by construction $\gamma_{0,n-1} \equiv_S \gamma'_{0,n-1}$, $\gamma_{n,p} \equiv_L \gamma'_{n,p}$ and $\gamma_{p+1,i} \equiv_S \gamma'_{p+1,i}$. However, $\gamma' \not\models c_{\bar{i}}$. Following the reasoning of proof of Theorem 2, we can conclude that we have found γ and γ' such that one leads to a secret state in Sec while the other does not. This gives a contradiction. □

Therefore we can state the following.

Proposition 2. *The property of a reaction system $\mathcal{A}$ to be n-p window i-state opaque with respect to a set of low level objects L and a secret sets of state Sec is decidable.*

If a system is 0-i window i-state opaque then it is i-current state opaque.

Example 5. Consider again the reaction system $\mathcal{A}_2$, L and Sec as in Example 4. $\mathcal{A}_2$ was 3-current state opaque. However, $\mathcal{A}_2$ it is not 3-3 window i-state opaque. Recall that

$$\mathcal{DNF}(\texttt{fbp}(C,3) \wedge \texttt{fbp}(E,3)) = (A_3 \wedge \neg B_3 \wedge D_2 \wedge F_3) \vee (A_3 \wedge \neg D_3 \wedge F_3).$$

Then, $\{A \mid A_j \in atom((A_3 \wedge \neg B_3 \wedge D_2 \wedge F_3)),$ with $3 \leq j \leq 3\} \cap (S \setminus L) = \emptyset$ and Theorem 3 is not satisfied. Consider, for example, we can choose $\gamma = \{\}, \{\}, \{D\}, \{A, B, F\}$, then any γ' such that $\gamma_{0,2} \equiv_S \gamma'_{0,2}$, $\gamma_{3,3} \equiv_L \gamma'_{3,3}$ must be $\gamma' = \{\}, \{\}, \{D\}, C'_3$ with $C'_3 \supseteq \{A, B, F\}$, therefore also γ' will lead to a secret state in Sec.

Finally, note that $\mathcal{A}_2$ is n-3 window i-state opaque for any $0 \leq n \leq 2$.

6 Conclusions and further work

In this paper we have defined two state based security properties, that are, Current State Opacity and n-p Window State Opacity for reaction systems. We proposed effectively computable methods for verifying such properties based on the new notion of formula based predictor for set of secret states sets, newly defined in Section 4.

As further work we plan to elaborate other notions of opacity for reaction systems. The first one is in a sense a complement notion to n-p Window i-State Opacity. We consider an observer who can see only a small "window" of computation. If after that computation a secret state has been reached we expect that there exists seemingly the same window which leads to non-secret states. Also we plan to study the notion Initial State Opacity. In this case an observer tries to learn properties of an initial state of the computation. We believe that these concepts, borrowed by the security theory, can be also studied in the context of reaction systems. Moreover, it would be interesting to study variants of reaction systems with a limited threshold assumption and with timed properties (for a process algebra example, see [16]).

References

1. A. Ehrenfeucht, G. Rozenberg, Reaction Systems, Fundam. Inform. 75 (1-4) (2007) 263–280.
2. R. Brijder, A. Ehrenfeucht, M. G. Main, G. Rozenberg, A Tour of reaction Systems, Int. J. Found. Comput. Sci. 22 (7) (2011) 1499–1517.
3. D. Gruska, R. Gori, P. Milazzo, Studying opacity of reaction systems through formula based predictors, in: Proc. of the 26th Int. Workshop on Concurrency, Specification and Programming, CS&P, 2017.
4. D. Gruska, R. Gori, P. Milazzo, Studying opacity of reaction systems through formula based predictors, Fundamenta Informaticae To appear.
5. J. A. Goguen, J. Meseguer, Security policies and security models, Proc. of IEEE Symposium on Security and Privacy.
6. J. Bryans, M. Koutny, P. Ryan, Modelling non-deducibility using petri nets, in: 2nd Workshop on Security Issues with Petri Nets and other Computational Models, 2004.
7. J. W. Bryans, M. Koutny, L. Mazaré, P. Y. Ryan, Opacity generalised to transition systems, International Journal of Information Security 7 (6) (2008) 421–435.
8. R. Brijder, A. Ehrenfeucht, G. Rozenberg, A Note on Causalities in Reaction Systems, ECEASST 30.
9. R. Barbuti, R. Gori, F. Levi, P. Milazzo, Investigating dynamic causalities in reaction systems, Theoretical Computer Science 623 (2016) 114–145.
10. R. Barbuti, R. Gori, F. Levi, P. Milazzo, Specialized predictor for reaction systems with context properties, in: Proc. of the 24th Int. Workshop on Concurrency, Specification and Programming, CS&P 2015, 2015, pp. 31–43.
11. R. Barbuti, R. Gori, F. Levi, P. Milazzo, Specialized predictor for reaction systems with context properties, Fundamenta Informaticae 147 (2-3) (2016) 173–191.
12. R. Barbuti, R. Gori, F. Levi, P. Milazzo, Generalized contexts for reaction systems: definition and study of dynamic causalities, Acta Inf. 55 (3) (2018) 227–267.
13. R. Barbuti, R. Gori, P. Milazzo, Multiset patterns and their application to dynamic causalities in membrane systems, in: Membrane Computing - 18th Int. Conference, CMC 2017, 2017, pp. 54–73.
14. R. Barbuti, R. Gori, P. Milazzo, Predictors for flat membrane systems, Theor. Comput. Sci. 736 (2018) 79–102.
15. R. Jacob, J. Lesage, J. Faure, Overview of discrete event systems opacity: Models, validation, and quantification, Annual Reviews in Control 41 (2016) 135–146.
16. M. C. Ruiz, D. Cazorla, F. Cuartero, J. J. Pardo, H. Macia, A bounded true concurrency process algebra for performance evaluation, in: FORTE Workshops 2004, 2007, pp. 143–155.

Preserving Behavior in Transition Systems from Event Structure Models*

Nataliya Gribovskaya[1] and Irina Virbitskaite[1,2]

[1] A.P. Ershov Institute of Informatics Systems, SB RAS,
6, Acad. Lavrentiev av., 630090 Novosibirsk, Russia
[2] Novosibirsk State University, 2, Pirogov av., 630090 Novosibirsk, Russia
{gribovskaya,virb}@iis.nsk.su

Abstract. Two structurally different methods of associating transition system semantics to event structure models are distinguished in the literature. One of them is based on configurations (event sets), the other on residuals (model fragments). In this paper, we consider three kinds of event structures (resolvable conflict structures, extended prime structures, stable structures), translate the other models into resolvable conflict structures and back, provide the isomorphism results on the two types of transition systems, and demonstrate the preservation of some bisimulations on them.

1 Introduction

Since the introduction of event structures in [26], many variants of event-oriented models have been proposed based on different behavioural relations between events and thus providing a different expressive power. Among the models are prime event structures [26] (with conjunctive[1] binary causality, represented by a partial order and being under the principle of finite causes, and symmetric irreflexive conflict, obeying the principle of conflict heredity; all these guarantee unique event enablings within the model); extended prime event structures [1] (with conjuctive binary causality, being possibly with cycles and not being under the principle of finite causes, and symmetric irreflexive conflict, not obeying the principle of conflict heredity; moreover, the relations can be overlapped); stable event structures [28] (with non-binary conjuctive causality, allowing for alternative enablings, and the stability constraint (i.e .the intersection of two non-conflicting causal predecessors sets for an event is a causal predecessors set for the event) resulting in unique enablings within a configuration); event structures for resolvable conflict [14] (with dynamic conflicts, i.e. conflicts can be resolved or created by the occurrences of other events), etc. Comparative analysis of some classes of event structures can be found in the works [1, 2, 11, 12, 14, 15, 16, 18].

Two methods of associating a labeled transition system [20] with an event-oriented model of a distributed system, such as an event structure [26] or a

* Supported by German Research Foundation through grant Be 1267/14-1.
[1] An event is enabled once all of its causal predecessors have occurred.

configuration structure [13], can be distinguished: a *configuration-based* and a *residual-based* method. In the first case,[2] states are understood as sets of events, called *configurations*, and state transitions are built by starting with the empty configuration and enlarging configurations by already executed events. In the second approach,[3] states are understood as event structures, and transitions are built by starting with the given event structure as an initial state and removing already executed parts thereof in the course of an execution.

In the literature, configuration-based transition systems seem to be predominantly used as the semantics of event structures, whereas residual-based transition systems are actively used in providing operational semantics of process calculi and in demonstrating the consistency of operational and denotational semantics. The two kinds of transition systems have occasionally been used alongside each other (see [18] as an example), but their general relationship has not been studied for a wide range of existing models. In a seminal paper, viz. [23], bisimulations between configuration-based and residual-based transition systems have been proved to exist for prime event structures [28]. The result of [23] has been extended in [5] to more complex event structure models with asymmetric conflict. Counterexamples illustrated that an isomorphism cannot be achieved with the various removal operators defined in [5, 23]. The paper [6] demonstrated that the operators can be tightened in such a way that *isomorphisms*, rather than just bisimulations, between the two types of transition systems belonging to a single event structure can be obtained. A key idea is to employ *non-executable (impossible)* events[4] if the model allows them (and to introduce a special non-executable event otherwise), in order to turn *model fragments* into parts of states. This idea has been applied by the authors on a wide variety of event structure models, and for a full spectrum of semantics (interleaving, step, pomset, multiset). Thanks to the results, a variety of facts known from the literature on configuration-based transition systems (e.g., [4, 10, 13, 28]) can be extended to residual-based ones.

The aim of this paper is to connect several models of event structures by providing behaviour preserving translations between them, and to demonstrate the retention of some of the bisimulation concepts in the two types of transition systems associated with the models under consideration.

In Section 2 of this paper, we consider three kinds of event structure models (resolvable conflict, extended prime, stable event structures), define removal operators for them, and translate the other models into resolvable conflict event structures and back. Section 3 contains the definitions of the two types of transition systems, describes the isomorphism results, and demonstrate the preservation of some bisimulations on the transition systems. Section 4 concludes.

[2] E.g., see [1, 2, 11, 12, 14, 15, 17, 18, 24, 27].
[3] E.g., see [3, 7, 8, 9, 18, 19, 21, 22, 25].
[4] In an event structure, an event is called non-executable or impossible if it does not occur in any configuration of the structure, i.e. the event is never executed.

2 Event Structure Models

2.1 Event Structures for Resolvable Conflict

In this section, we consider event structures for resolvable conflict, which were put forward in [14] to give semantics to general Petri nets. A structure for resolvable conflict consists of a set of events and an enabling relation $\vdash$ between sets of events. The enabling $X \vdash Y$ with sets X and Y imposes restrictions on the occurrences of events in Y by requiring that for all events in Y to occur, their *causes* – the events in X – have to occur before. This allows for modeling the case when a and b cannot occur together until c occurs, i.e., initially a and b are in conflict until the occurrence of c resolves this conflict. Notice that the event structure classes under consideration in this paper are unable to model the phenomena of resolvable conflict. In resolvable conflict structures, the enabling relation can also model conflicts: events from a set Y are *in irresolvable conflict* iff there is no enabling of the form $X \vdash Y$ for any set X of events. Further, an event can be impossible (i.e. non-executable in any system's run) if it has no enabling or has infinite causes or has impossible causes/prececessors.

Definition 1. *An* event structure for resolvable conflict (*RC*-structure) over L *is a tuple* $\mathcal{E} = (E, \vdash, L, l)$*, where* E *is a set of events;* $\vdash \subseteq \mathcal{P}(E) \times \mathcal{P}(E)$ *is the* enabling relation*;* L *is a set of labels;* $l : E \to L$ *is a labeling function.*

Let $\mathcal{E}$ be an *RC*-structure over L. For $X \subseteq E$ and $e \in E$, $Con(X) \iff \forall \widehat{X} \subseteq X \colon \exists Z \subseteq E \colon Z \vdash \widehat{X}$; $fCon(X) \iff X$ is finite and $Con(X)$. The *conflict relation* $\sharp \subseteq E \times E$ is given by: $d \sharp e \iff d \neq e \wedge \neg Con(\{d, e\})$. The *direct causality relation* $\prec \subseteq E \times E$ is defined as follows: $d \prec e \iff \forall X \subseteq E \colon (X \vdash e \Rightarrow d \in X)$. A set $X \subseteq E$ is *left-closed* iff X is finite, and for all $\widetilde{X} \subseteq X$ there exists $\widehat{X} \subseteq X$ such that $\widehat{X} \vdash \widetilde{X}$. The set of the left-closed sets of $\mathcal{E}$ is denoted as $LC(\mathcal{E})$. Clearly, any left-closed set is conflict-free. Let $Conf(\mathcal{E}) = \{\{e_1, \ldots, e_n\} \subseteq E \mid n \geq 0, \forall i \leq n \colon \forall X \subseteq \{e_1, \ldots, e_i\} \colon \exists Y \subseteq \{e_1, \ldots, e_{i-1}\} \colon Y \vdash X\}$ be the set of configurations of $\mathcal{E}$. Clearly, any configuration X is a left-closed set but not conversely.

Consider some properties of resolvable conflict event structures.

Definition 2. *An RC-structure* $\mathcal{E} = (E, \vdash, L, l)$ *is called*

- rooted *iff* $(\emptyset, \emptyset) \in \vdash$*;*
- pure *iff* $X \vdash Y \Rightarrow X \cap Y = \emptyset$*;*
- singular *iff* $X \vdash Y \Rightarrow X = \emptyset \vee \mid Y \mid = 1$*;*
- manifestly conjunctive *iff there is at most one* X *with* $X \vdash Y$*, for all* $Y \subseteq E$*;*
- conjuctive *iff* $X_i \vdash Y (i \in I \neq \emptyset) \Rightarrow \bigcap_{i \in I} X_i \vdash Y$*;*
- locally conjuctive *iff* $X_i \vdash Y (i \in I \neq \emptyset) \wedge Con(\bigcup_{i \in I} X_i \cup Y) \Rightarrow \bigcap_{i \in I} X_i \vdash Y$*;*
- with finite causes *iff* $X \vdash Y \Rightarrow X is finite$*;*
- with binary conflict *iff* $\mid X \mid > 2 \Rightarrow \emptyset \vdash X$*;*
- in the standard form *iff* $\vdash = \{(A, B) \mid A \cap B = \emptyset,\ A \cup B \in LC(\mathcal{E})\}$*;*
- 2-coherent *iff* $X \cup Y \in LC(\mathcal{E})$*, for all* $X, Y \in LC(\mathcal{E})$ *s.t.* $X \cup Y \subseteq Z \in LC(\mathcal{E})$.[5]

[5] This property is useful when proving Theorem 1.

Lemma 1. *An RC-structure $\mathcal{E} = (E, \vdash, L, l)$ can be transformed into:*

- *a pure RC-structure $PU(\mathcal{E}) = (E, \widehat{\vdash}, L, l)$[6] s.t. $Conf(\mathcal{E}) = Conf(PU(\mathcal{E}))$, if $\mathcal{E}$ is a singular RC-structure;*
- *an RC-structure $SF(\mathcal{E}) = (E, \widetilde{\vdash}, L, l)$[7] in the standard form s.t. $LC(\mathcal{E}) = LC(SF(\mathcal{E}))$. Moreover, $Conf(\mathcal{E}) = Conf(SF(\mathcal{E}))$, if $\mathcal{E}$ is a pure RC-structure.*

Example 1. As an example, consider the pure, manifestly conjuctive, non-singular, non-2-coherent *RC*-structure $\mathcal{E}^{rc} = (E^{rc}, \vdash^{rc}, L, l^{rc})$ with finite causes and binary conflict from [15], where $E^{rc} = \{a, b, c\}$; $\vdash^{rc}$ consists of $\emptyset \vdash X$ for all $X \neq \{a, b\}$ and $\{c\} \vdash \{a, b\}$; $L = E^{rc}$; and l^{rc} is the identity labeling function. It is easy to see that $LC(\mathcal{E}^{rc}) = Conf(\mathcal{E}^{rc}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$. This *RC*-structure models the initial conflict between the events a and b that can be resolved by the occurrence of the event c. The structure $\mathcal{E}^{rc}$ can be presented in the standard form $\widetilde{\mathcal{E}}^{rc}$ with $\widetilde{\vdash}^{rc}$ consisting of $A \widetilde{\vdash} B$ such that $B \subseteq C \in LC(\mathcal{E})$ and $A = C \setminus B$, i.e. $\widetilde{\vdash}^{rc} = \{(\emptyset, \emptyset), (\emptyset, \{a\}), (\{a\}, \emptyset), (\emptyset, \{b\}), (\{b\}, \emptyset), (\emptyset, \{c\}), (\{c\}, \emptyset), (\emptyset, \{a, c\}), (\{a, c\}, \emptyset), (\{a\}, \{c\}), (\{c\}, \{a\}), \ldots, (\emptyset, \{a, b, c\}), (\{a, b, c\}, \emptyset)\}$.

The standard form of *RC*-structures and the ability to specify impossible events in the model allows for developing a relatively simple structural definition of a removal operator which is necessary for residual semantics.

Definition 3. *For an RC-structure $\mathcal{E} = (E, \vdash, L, l)$ in the standard form and $X \in LC(\mathcal{E})$, a removal operator is defined as follows: $\mathcal{E} \setminus X = (E', \vdash', L, l')$, where*

$$E' = E \setminus X$$
$$\vdash' = \{(A', B') \mid \exists (A, B) \in \vdash \text{ s.t. } A' = A \cap E', B' = B \cap E', (A' \cup B' \cup X) \in LC(\mathcal{E})\}$$
$$l' = l \mid_{E'}$$

According to the definition above, all the events in X are removed; however, we retain the events, not forming left-closed sets with the events in X and hence conflicting with some events in X, making the retained events impossible by deleting their enabling relations.

From now on, we use $\mathbb{E}^{rc}_L$ to denote the class of rooted, singular, locally conjuctive *RC*-structures with binary conflict.

2.2 Extended Prime Event Structures

For reasons of flexibility, the authors of [1] propose to generalise ordinary prime event structures [28][8] by dropping the transitivity and acyclicity of causality,

[6] An *RC*-structure $PU(\mathcal{E}) = (E, \widehat{\vdash}, L, l)$ can be directly obtained by putting $\widehat{\vdash} = \vdash \setminus \{(A, B) \in \vdash \mid \emptyset \neq B \subseteq A\}$.

[7] An *RC*-structure $SF(\mathcal{E}) = (E, \widetilde{\vdash}, L, l)$ can be directly obtained by putting $\widetilde{\vdash} = \{(A, B) \mid B \subseteq C \in LC(\mathcal{E}),\ A = C \setminus B\}$.

[8] A labeled prime event structure over a set L of actions is a tuple $\mathcal{E} = (E, \sharp, \leq, L, l)$, where E is a set of events; $\leq \subseteq E \times E$ is a partial order (the *causality relation*),

as well as the principles of finite causes and conflict inheritance.[9] As opposed to prime event structures, the extended version allows for impossible events. In this model, events can be impossible because of enabling cycles, or an overlapping between the enabling and the conflict relation, or because of impossible causes/predecessors.

Definition 4. *An* extended prime event structure (EP-structure) over L *is a triple* $\mathcal{E} = (E, \sharp, \prec, L, l)$*, where* E *is a set of events;* $\sharp \subseteq E \times E$ *is an irreflexive symmetric relation (the* conflict relation*);* $\prec \subseteq E \times E$ *is the* enabling relation*;* L *is a set of labels;* $l : E \to L$ *is a labeling function. Let* $\mathbb{E}_L^{ep}$ *denote the class of* EP*-structures over* L*.*

Let $\mathcal{E} = (E, \sharp, \prec, L, l)$ be an EP-structure. For $e \in E$, define $\downarrow e$ as a maximal subset of E such that $\forall e' \in \downarrow e : e' \prec e$. For $X \subseteq E$, let $\sharp(X) = \{e' \in E \mid \exists e \in X : e \sharp e'\}$. We call a set $X \subseteq E$ a *configuration* of $\mathcal{E}$ if X is finite, conflict-free (i.e. $\forall e, e' \in X : \neg(e \sharp e')$), left-closed (i.e. $\forall e, e' \in E : e \prec e' \wedge e' \in X \Rightarrow e \in X$), and does not contain enabling cycles (i.e., $\nexists e_1, \ldots, e_n \in X : e_1 \prec \ldots \prec e_n \prec e_1$ $(n \geq 1)$). The set of configurations of $\mathcal{E}$ is denoted by $Conf(\mathcal{E})$.

In the graphical representation of an EP-structure, pairs of events related by the enabling relation are connected by arrows; pairs of the events included in the conflict relation are marked by the symbol $\sharp$.

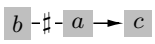
$\mathcal{E}^{ep}$: 

Fig. 1. An extended prime event structure $\mathcal{E}^{ep}$

Example 2. Figure 1 depicts the EP-structure $\mathcal{E}^{ep}$ over $L = \{a, b, c\}$, with $E^{ep} = L$; $\sharp^{ep} = \{(a, b), (b, a)\}$; $\prec^{ep} = \{(a, c)\}$; and the identity labeling function l^{ep}. Observe that the principle of conflict inheritance is violated. The set of configurations of $\mathcal{E}^{ep}$ is $\{\emptyset, \{a\}, \{b\}, \{a, c\}\}$.

Consider the definition of the removal operator for EP-structures.

Definition 5. *For* $\mathcal{E} \in \mathbb{E}_L^{ep}$ *and* $X \in Conf(\mathcal{E})$*, a removal operator is defined as follows:* $\mathcal{E} \setminus X = (E', \prec', \sharp', L, l')$*, with*

$$\begin{array}{lcl} E' & = & E \setminus X \\ \sharp' & = & \sharp \cap (E' \times E') \\ \prec' & = & (\prec \cap (E' \times E')) \ \cup \ \{(e, e) \mid e \in \sharp(X)\} \\ l' & = & l \mid_{E'} \end{array}$$

satisfying the *principle of finite causes*: $\forall e \in E : \lfloor e \rfloor = \{e' \in E \mid e' \leq e\}$ is finite; $\sharp \subseteq E \times E$ is an irreflexive and symmetric relation (the *conflict relation*), satisfying the *principle of hereditary conflict*: $\forall e, e', e'' \in E : e \leq e'$ and $e \sharp e''$ then $e' \sharp e''$; and $l : E \to L$ is a labeling function.

[9] It was noted in [1] that, as far as finite configurations are concerned, this does not lead to an increase in expressive power.

We see that the events in X are removed, yielding a reduction of the enabling and conflict relations. At the same time, any event conflicting with some event in X is retained, equipping it with an enabling cycle, thereby making the conflicting event impossible.

Translate EP-structures into RC-structures and conversely. For an EP-structure $EP = (E, \sharp, \prec, L, l)$, define $\mathcal{RC}(EP) = (E' = E, \vdash', L, l = l')$, where

$$X \vdash' Y \iff \begin{cases} \text{either } Y = \{e\}, X = \downarrow e, \\ \text{or } Y = \{e, e'\}, e \neq e', \neg(e \sharp e'), X = \emptyset, \\ \text{or } \mid Y \mid \neq 1, 2, X = \emptyset. \end{cases}$$

For an RC-structure $RC = (E', \vdash', L, l')$, let $\mathcal{EP}(RC) = (E'' = E', \sharp'' = \sharp', \prec'' = \prec', L, l'' = l')$.

Lemma 2. *(i) For EP an EP-structure, $\mathcal{RC}(EP)$ is a rooted, singular, manifestly conjuctive RC-structure with binary conflict such that $Conf(EP) = Conf(\mathcal{RC}(EP))$.*

(ii) For RC a rooted, singular, conjuctive RC-structure with binary conflict, $\mathcal{EP}(RC)$ is an EP-structure such that $Conf(RC) = Conf(\mathcal{EP}(RC))$.

2.3 Stable Event Structures

Stable event structures, introduced in the work of Winskel [27] in order to overcome the unique enabling problem of prime event structures, have an enabling relation indicating which (usually finite) sets X of events are possible prerequisites of a single event e, written $X \vdash e$. We consider the version of stable event structures of [28] where the conflict relation is specified for sets with two events.

Definition 6. *A* stable event structure over L (S-structure) *is a tuple $\mathcal{E} = (E, \sharp, \vdash, L, l)$, where*

- *E is a set of events;*
- *$\sharp \subseteq E \times E$ is an irreflexive, symmetric relation (the* conflict relation*). We shall write Con for the set of finite conflict-free subsets of E, i.e. those finite subsets $X \subseteq E$ for which $\forall e, e' \in X \ : \ \neg(e \sharp e')$. $X \in Con$ means that the events in X could happen in the same run, i.e. they are consistent;*
- *$\vdash \subseteq Con \times E$ is the* enabling relation *which satisfies $X \vdash e$ and $X \subseteq Y \in Con \Rightarrow Y \vdash e$; and, moreover, $X \vdash e$, $Y \vdash e$, and $X \cup Y \cup \{e\} \in Con \Rightarrow X \cap Y \vdash e$ (*the stability principle*). $\vdash$ indicates possible causes: an event e can occur whenever for some X with $X \vdash e$ all events in X have occurred before. The minimal enabling relation $\vdash_{min}$ is defined as follows: $X \vdash_{min} e$ iff $X \vdash e$ and for all $Y \subseteq X$ if $Y \vdash e$ then $Y = X$;*
- *L is a set of actions;*
- *$l : E \to L$ is a labeling function.*

Let $\mathbb{E}^s_L$ denote the class of S-structures over L.

A set $X \subseteq E$ is a *configuration* of an S-structure $\mathcal{E}$ iff X is finite, conflict-free (i.e., $X \in Con$), and secured (i.e., there are $e_1, \ldots, e_n$ such that $X = \{e_1, \ldots, e_n\}$

and $\{e_1, \ldots, e_i\} \vdash e_{i+1}$, for all $i < n$). The set of configurations of $\mathcal{E}$ is denoted $Conf(\mathcal{E})$. For an S-structure $\mathcal{E}$, $X \in Conf(\mathcal{E})$, and $e, e' \in X$, let $e' \prec_X e$ iff e' belongs to the smallest subset Y of X with $Y \vdash e$.

Example 3. Consider the S-structure $\mathcal{E}^s$ over $L = \{a, b, c, d\}$, with $E^s = L$; $\sharp^s = \{(a, b), (b, a)\}$; $\vdash^s_{min} = \{(\emptyset, a), (\emptyset, b), (\emptyset, c), (\{a\}, d), (\{b, c\}, d)\}$; and the identity labeling function l^s. The set of configurations of $\mathcal{E}^s$ is $\{\emptyset, \{a\}, \{b\}, \{c\}, \{a, c\}, \{b, c\}, \{a, d\}, \{a, c, d\}, \{b, c, d\}\}$. Notice that $\mathcal{E}^s$ is not a flow event structure because the event c not conflicting with the event a may be a cause for d or may not.

Definition 7. *For $\mathcal{E} = (E, \sharp, \vdash, L, l) \in \mathbb{E}^s_L$ and $X \in Conf(\mathcal{E})$, a removal operator is defined as follows: $\mathcal{E} \setminus X = (E', \sharp', \vdash', L, l')$, with*

$$
\begin{array}{lcl}
E' & = & E \setminus X \\
\sharp' & = & \sharp \cap (E' \times E') \\
\vdash' & = & \{(W', e) \mid W' \in Con',\ \exists (W'', e) \in \vdash'_{min} \ \ s.t.\ W'' \subseteq W'\} \ \textit{where} \\
 & & \quad \vdash'_{min} = \{(W'', e) \mid \exists (W, e) \in \vdash_{min} \ \ s.t.\ W'' = W \cap E',\ e \in E', \\
 & & \qquad\qquad\qquad\qquad\qquad\qquad W'' \cup X \in Con,\ \{e\} \cup X \in Con\} \\
l' & = & l\,|_{E'}
\end{array}
$$

We see that all the events in X are deleted; the conflict relation $\sharp'$ contains the pairs of remaining conflicting events; the definition of $\vdash'$ is based on that of $\vdash'_{min}$, which consists of the pairs from $\vdash$ without the pairs whose events conflict with some event in X, thereby making them impossible.

For an S-structure $S = (E, \sharp, \vdash, L, l)$, let $\mathcal{RC}(G) = (E' = E, \vdash', L, l')$, where

$$
X \vdash' Y \iff \begin{cases} \text{either } Y = \{e\}, e \in E, X \vdash e, \\ \text{or } \mid Y \mid = 2, Y \in Con, X = \emptyset, \\ \text{or } \mid Y \mid \neq 1, 2, X = \emptyset. \end{cases}
$$

For an RC-structure $RC = (E', \vdash', L, l')$, let $\mathcal{S}(RC) = (E'' = E', \sharp'' = \sharp', \vdash'', L, l'')$, where $X \vdash'' e \iff e \in E',\ X \subseteq E',\ fCon'(X)$, and $\exists Y \subseteq X : Y \vdash' \{e\}$.

Lemma 3. *[15]*

(i) For S an S-structure, $\mathcal{RC}(S)$ is a rooted, singular, locally conjuctive RC-structure with finite causes and binary conflict s.t. $Conf(S) = Conf(\mathcal{RC}(F))$.

(ii) For RC a rooted, singular, locally conjuctive RC-structure with finite causes and binary conflict, $\mathcal{S}(RC)$ is an S-structure s.t. $Conf(RC) = Conf(\mathcal{S}(RC))$.

2.4 Different Semantics

In this subsection, we define different semantics for the event structure models under consideration. From now on, we treat $\mathcal{E}$ as an event structure over L specified in Definitions 1, 4, and 6, if not defined otherwise. Moreover, let $\mathbb{E}_L = \mathbb{E}^{ep}_L \cup \mathbb{E}^s_L \cup \mathbb{E}^{rc}_L$.

We first introduce auxiliary notations. Given configurations $X, X' \in Conf(\mathcal{E})$, we write:

- $X \rightarrow_{int} X'$ iff $X \subseteq X'$ and $X' \setminus X = \{e\}$;
- $X \rightarrow_{step} X'$ iff $X \subseteq X'$ and $X'' \in \mathit{Conf}(\mathcal{E})$, for all $X \subseteq X'' \subseteq X'$;
- $X \rightarrow_{pom} X'$ iff $X \subseteq X'$ and $\leq_{X' \setminus X}$ is a partial order;
- $X \rightarrow_{whp} X'$ iff $X \subseteq X'$ and $\leq_{X'}$ is a partial order.

For $\star \in \{int, step, pom\}$, a configuration $X \in \mathit{Conf}(\mathcal{E})$ is a *configuration in $\star$-semantics* of $\mathcal{E}$ iff $\emptyset \rightarrow_\star^* X$, where $\rightarrow_\star^*$ is the reflexive and transitive closure of $\rightarrow_\star$. Let $\mathit{Conf}_\star(\mathcal{E})$ denote the set of configurations in $\star$-semantics of $\mathcal{E}$.

Lemma 4. *Given an event structure $\mathcal{E} \in \mathbb{E}_L$ and $\star, * \in \{int, step, pom\}$,*

(i) for a configuration $X \in \mathit{Conf}(\mathcal{E})$, the transitive and reflexive closure of $\prec_X$, $\leq_X$, is a partial order. Let $\mathcal{E}\lceil X = (X, \leq_X, L, l\mid_X)$;
(ii) $\mathit{Conf}(\mathcal{E}) = \mathit{Conf}_\star(\mathcal{E}) = \mathit{Conf}_*(\mathcal{E})$.

Given $\star \in \{int, step, pom\}$, an event structure $\mathcal{E}$ over L, and configurations $X, X' \in \mathit{Conf}_\star(\mathcal{E})$ such that $X \rightarrow_\star X'$, we write:

- $l_{int}(X' \setminus X) = a$ iff $X' \setminus X = \{e\}$ and $l(e) = a$, if $\star = int$;
- $l_{step}(X' \setminus X) = M$ iff $M(a) = |\{e \in X' \setminus X \mid l(e) = a\}|$, for all $a \in L$, if $\star = step$;
- $l_{pom}(X' \setminus X) = \mathcal{Y}$ iff $\mathcal{Y} = [(X' \setminus X, \leq_{X'} \cap (X' \setminus X \times X' \setminus X), L, l\mid_{X' \setminus X})]$, if $\star = pom$;
- $l_{whp}(X) = \mathcal{Y}$ iff $\mathcal{Y} = [(X, \leq_X, L, l\mid_X)]$.

Let $\mathcal{E}$ be an event structure over L and $X = \{e_1, \ldots, e_n\} \in \mathit{Conf}_{int}(\mathcal{E})$ $(n \geq 0)$. We call $e_1 \ldots e_n$ a *derivation* of X iff $X_0 = \emptyset \rightarrow_{int} X_1 \ldots X_{n-1} \rightarrow_{int} X_n = X$, and $X_i \setminus X_{i-1} = \{e_i\}$, for all $1 \leq i \leq n$. A derivation $e_1 \ldots e_n$ of $X \in \mathit{Conf}_{int}(\mathcal{E})$ and a derivation $f_1 \ldots f_n$ of $X' \in \mathit{Conf}_{int}(\mathcal{E}')$ are *equal* (denoted $e_1 \ldots e_n \sim f_1 \ldots f_n$) iff there is an isomorphism $\iota : \mathcal{E}\lceil X \rightarrow \mathcal{E}'\lceil X'$ with $\iota(e_1 \ldots e_n) := \iota(e_1) \ldots \iota(e_n) = f_1 \ldots f_n$. Let $\mathit{Der}(X)$ denote the set of all equivalence classes $[e_1 \ldots e_n]$ of derivations of X. For $[e_1 \ldots e_n] \in \mathit{Der}(X)$, define $l_{hp}([e_1 \ldots e_n]) := a_1 \ldots a_n$, where $l_i(e_i) = a_i$ $(1 \leq i \leq n)$.

3 Transition Systems $TC(\mathcal{E})$ and $TR(\mathcal{E})$

3.1 Definitions and Comparisons

In this subsection, we first give some basic definitions concerning labeled transition systems, and then define the mappings $TC(\mathcal{E})$ and $TR(\mathcal{E})$, which associate two distinct kinds of transition systems – one whose states are configurations and one whose states are residual event structures – with an event structure $\mathcal{E}$ over L.

A transition system $T = (S, \rightarrow, i)$ over a set $\mathcal{L}$ of labels consists of a set of states S, a transition relation $\rightarrow \subseteq S \times \mathcal{L} \times S$, and an initial state $i \in S$. Two transition systems over $\mathcal{L}$ are *isomorphic* if their states can be mapped one-to-one to each other, preserving transitions and initial states. We call a relation $R \subseteq S \times S'$ a *bisimulation* between transition systems T and T' over $\mathcal{L}$ iff

$(i, i') \in R$, and for all $(s, s') \in R$ and $l \in \mathcal{L}$: if $(s, l, s_1) \in \rightarrow$, then $(s', l, s'_1) \in \rightarrow$ and $(s_1, s'_1) \in R$, for some $s'_1 \in S'$; and if $(s', l, s'_1) \in \rightarrow$, then $(s, l, s_1) \in \rightarrow$ and $(s_1, s'_1) \in R$, for some $s_1 \in S$.

Introduce additional auxiliary notations. For a fixed set L of labels of event structures, define $\mathbb{L}_{int} := L$, $\mathbb{L}_{pom} := Pom_L$ (the set of isomorphic classes of partial orders labeled over L), and $\mathbb{L}_{Der} := L^*$, being another sets of labels of the transition systems.

We are ready to define labeled transition systems with configurations as states.

Definition 8. *For an event structure $\mathcal{E}$ over L and $\star \in \{int, step, pom\}$,*

- *$TC_\star(\mathcal{E})$ is the transition system $(Conf_\star(\mathcal{E}), \multimap_\star, \emptyset)$ over $\mathbb{L}_\star$, where $X \stackrel{p}{\multimap_\star} X'$ iff $X \rightarrow_\star X'$ and $p = l_\star(X' \setminus X)$;*
- *$TC_{whp}(\mathcal{E})$ is the transition system $(Conf_{int}(\mathcal{E}), \multimap_{whp}, \emptyset)$ over $\mathbb{L}_{pom}$, where $X \stackrel{p}{\multimap_{whp}} X'$ iff $X \rightarrow_{whp} X'$ and $p = l_{whp}(X')$;*
- *$TC_{hp}(\mathcal{E})$ is the transition system $(\{Der(X) \mid X \in Conf_{int}(\mathcal{E})\}, \multimap_{hp}, \epsilon)$ over $\mathbb{L}_{Der}$, where $[e_1 \ldots e_n] \stackrel{q}{\multimap_{hp}} [e_1 \ldots e_n e_{n+1}]$ $(n \geq 0)$ iff $\{e_1, \ldots, e_n\}$, $\{e_1, \ldots, e_n, e_{n+1}\} \in Conf_{int}(\mathcal{E})$, and $q = l_{hp}([e_1 \ldots e_n e_{n+1}])$.*

Consider the definition of labeled transition systems with residuals as states.

Definition 9. *For an event structure $\mathcal{E}$ over L and $\star \in \{int, step, pom\}$,*

- *$Reach_\star(\mathcal{E}) = \{\mathcal{F} \mid \exists \mathcal{E}_0, \ldots, \mathcal{E}_k$ $(k \geq 0)$ s.t. $\mathcal{E}_0 = \mathcal{E}$, $\mathcal{E}_k = \mathcal{F}$, and $\mathcal{E}_i \multimap_\star^X \mathcal{E}_{i+1}$ $(i < k)\}$, where $\mathcal{E}_i \multimap_\star^X \mathcal{E}_{i+1}$ iff $\exists X \in Conf_\star(\mathcal{E}_i)$: $\mathcal{E}_{i+1} = \mathcal{E}_i \setminus X$ and $\emptyset \rightarrow_\star X$ in $\mathcal{E}_i$;*
- *$TR_\star(\mathcal{E})$ is the transition system $(Reach_\star(\mathcal{E}), \multimap_\star, \mathcal{E})$ over $\mathbb{L}_\star$, where $\mathcal{F} \stackrel{p}{\multimap_\star} \mathcal{F}'$ iff $\exists X \in Conf_\star(\mathcal{F})$: $\mathcal{F} \multimap_\star^X \mathcal{F}'$ and $p = l_\star(X)$;*
- *$TR_{whp}(\mathcal{E})$ is the transition system $(Reach_{int}(\mathcal{E}), \multimap_{whp}, \mathcal{E})$ over $\mathbb{L}_{pom}$, where $\mathcal{F} \stackrel{p}{\multimap_{whp}} \mathcal{F}'$ iff $\exists X, X' \in Conf_{int}(\mathcal{E})$: $\mathcal{F} = \mathcal{E} \setminus X, \mathcal{F}' = \mathcal{E} \setminus X'$, $X \multimap_{whp} X'$, and $p = l_{whp}(X')$;*
- *$TR_{hp}(\mathcal{E})$ is the transition system $(Reach_{int}(\mathcal{E}), \multimap_{hp}, \mathcal{E})$ over $\mathbb{L}_{pom}$, where $\mathcal{F} \stackrel{q}{\multimap_{hp}} \mathcal{F}'$ iff $\exists X, X' \in Conf_{int}(\mathcal{E})$: $\mathcal{F} = \mathcal{E} \setminus X$, $\mathcal{F}' = \mathcal{E} \setminus X'$, $[e_1 \ldots e_n] \stackrel{q}{\multimap_{hp}} [e_1 \ldots e_n e_{n+1}]$, where $[e_1 \ldots e_n] \in Der(X)$, $[e_1 \ldots e_n e_{n+1}] \in Der(X')$, and $q = l([e_1 \ldots e_n e_{n+1}])$.*

For instance, Figures 2–4 indicate the transition systems $TR_\star(\mathcal{E})$ with the states — the residuals of the structures considered in Examples 1–3, respectively. Here, $\star = step$, if $\mathcal{E} = \mathcal{E}^{rc}$; $\star = whp$, if $\mathcal{E} = \mathcal{E}^{ep}$; and $\star = pom$, if $\mathcal{E} = \mathcal{E}^s$.

Theorem 1. *Given $\star \in \{int, step, pom, whp\}$, $TC_\star(\mathcal{E})$ and $TR_\star(SF(PU(\mathcal{E})))$ $(TR_\star(\mathcal{E}))$ are isomorphic; however, $TC_{hp}(\mathcal{E})$ and $TR_{hp}(SF(PU(\mathcal{E})))$ $(TR_{hp}(\mathcal{E}))$ are not bisimilar; where $\mathcal{E} \in \mathbb{E}_L^{rc}$ $(\mathcal{E} \in \mathbb{E}_L^{ep} \cup \mathbb{E}_L^s)$.*

It is easy to see that even for the EP-structure $\mathcal{E}_1^{ep}$ over $L = \{a, b, c\}$, with $E_1^{ep} = L$;, $\sharp_1^{ep} = \emptyset$, $\rightarrow_1^{ep} = \{(a, c), (b, c)\}$, and the identity labeling function l_1^{ep}, $TC_{hp}(\mathcal{E}_1^{ep})$ and $TR_{hp}(\mathcal{E}_1^{ep})$ are not bisimilar.

From Lemmas 1, 2, 3, and Theorem 1 we get

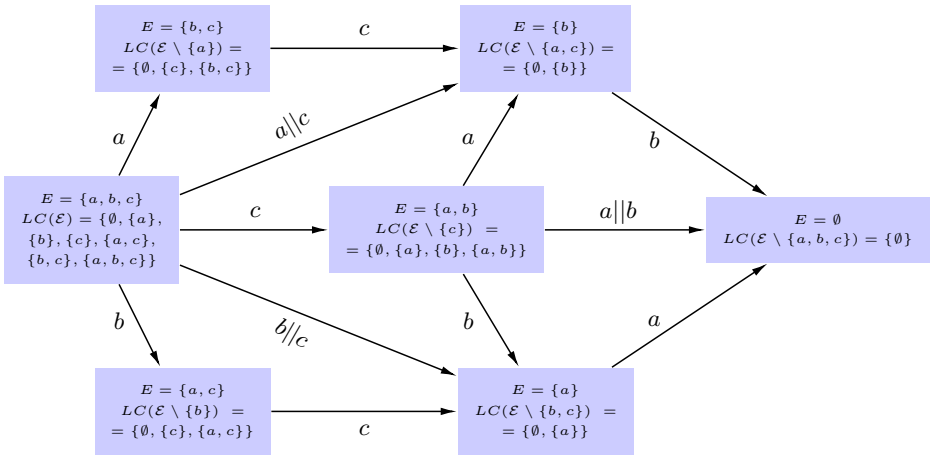


Fig. 2. The residual transition system $TR_{step}(\mathcal{E}^{rc})$

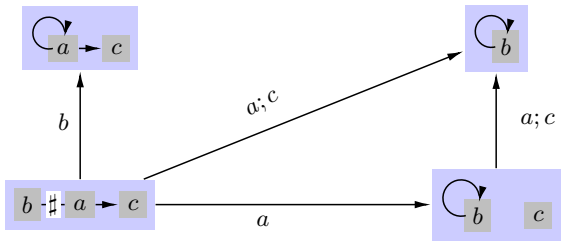


Fig. 3. The residual transition system $TR_{whp}(\mathcal{E}^{ep})$

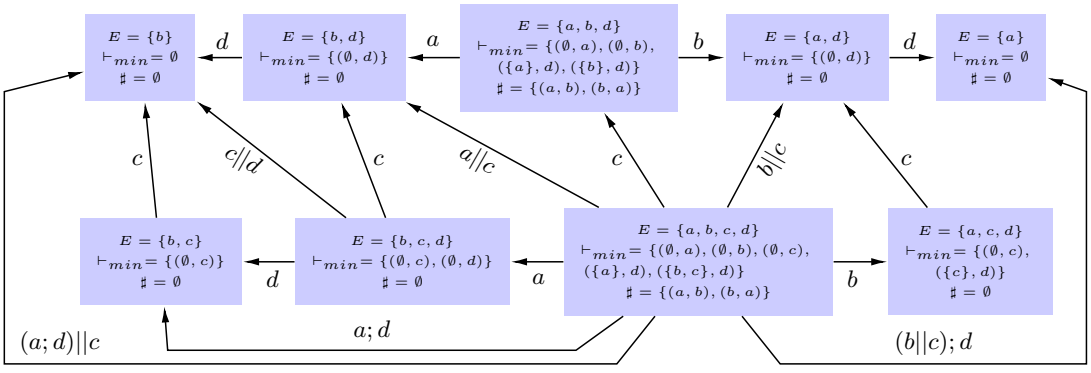


Fig. 4. The residual transition system $TR_{pom}(\mathcal{E}^{s})$

Corollary 1. *Given* $\star \in \{int, step, pom, whp\}$,

(i) $TR_\star(\mathcal{E})$ *and* $TR_\star(SF(PU(\mathcal{RC}(\mathcal{E}))))$ *are isomorphic, if* $\mathcal{E} \in \mathbb{E}_L^{ep} \cup \mathbb{E}_L^s$;
(ii) $TR_\star(SF(PU(\mathcal{E})))$, $TR_\star(\mathcal{EP}(\mathcal{E}))$, *and* $TR_\star(\mathcal{S}(\mathcal{E}))$ *are isomorphic, if* $\mathcal{E} \in \mathbb{E}_L^{rc}$.

3.2 Preserving Bisimulations by the Operators $TC(\cdot)$ and $TR(\cdot)$

We first introduce bisimulation concepts on the event structure models.

Event structures $\mathcal{E}$ and $\mathcal{E}'$ from $\mathbb{E}_L$ are *interleaving, step, pomset, respectively, bisimilar* iff $TC(\mathcal{E}_\star)$ and $TC(\mathcal{E}'_\star)$ are bisimilar for $\star \in \{int, step, pom\}$, respectively. For event structures $\mathcal{E}$ and $\mathcal{E}'$ over L,

- a relation $R \subseteq Conf_{int}(\mathcal{E}) \times Conf_{int}(\mathcal{E}')$ is called *weak history preserving bisimulation* iff $(\emptyset, \emptyset) \in R$ and for any $(X, Y) \in R$ it holds:
  - there is an isomorphism between $\mathcal{E}\lceil X$ and $\mathcal{E}'\lceil Y$;
  - if $X \subseteq X'$ for some $X' \in Conf_{int}(\mathcal{E})$, then $Y \subseteq Y'$ for some $Y' \in Conf_{int}(\mathcal{E}')$ such that $(X', Y') \in R$;
  - if $Y \subseteq Y'$ for some $Y' \in Conf_{int}(\mathcal{E}')$, then $X \subseteq X'$ for some $X' \in Conf_{int}(\mathcal{E})$ such that $(X', Y') \in R$.
- a relation R consisting of triples (X, f, Y), where $X \in Conf_{int}(\mathcal{E})$, $Y \in Conf_{int}(\mathcal{E}')$, and $f : \mathcal{E}\lceil X \to \mathcal{E}'\lceil Y$ is an isomorphism, is called *history preserving bisimulation* iff $(\emptyset, \emptyset, \emptyset) \in R$ and for any $(X, f, Y) \in R$ it holds:
  - if $X \subseteq X'$ for some $X' \in Conf_{int}(\mathcal{E})$, then $Y \subseteq Y'$ for some $Y' \in Conf_{int}(\mathcal{E}')$ such that $f' \mid_X = f$ for some isomorphism $f' : X' \to Y'$, and $(X', f', Y') \in R$;
  - if $Y \subseteq Y'$ for some $Y' \in Conf_{int}(\mathcal{E}')$, then $X \subseteq X'$ for some $X' \in Conf_{int}(\mathcal{E})$ such that $f' \mid_X = f$ for some isomorphism $f' : X' \to Y'$, and $(X', f', Y') \in R$.

Theorem 2. *Given* $\mathcal{E}, \mathcal{E}' \in \mathbb{E}_L$, $\mathcal{E}$ *and* $\mathcal{E}'$ *are weak history preserving bisimilar iff* $TC_{whp}(\mathcal{E})$ *and* $TC_{whp}(\mathcal{E}')$ *are bisimilar;* $\mathcal{E}$ *and* $\mathcal{E}'$ *are history preserving bisimilar iff* $TC_{hp}(\mathcal{E})$ *and* $TC_{hp}(\mathcal{E}')$ *are bisimilar.*

Corollary 2. $\mathcal{E}$ *and* $\mathcal{E}'$ *are interleaving, step, pomset, weak history preserving, respectively, bisimilar iff* $TR_\star(ST(PU(\mathcal{E})))$ *and* $TR_\star(ST(PU(\mathcal{E}')))$ *(*$TR_\star(\mathcal{E})$ *and* $TR_\star(\mathcal{E}')$*) are bisimilar for* $\star \in \{int, step, pom, whp\}$, *respectively, where* $\mathcal{E}, \mathcal{E}' \in \mathbb{E}_L^{rc}$ *(*$\mathcal{E}, \mathcal{E}' \in \mathbb{E}_L^{ep} \cup \mathbb{E}_L^s$*).*

4 Concluding Remarks

In this paper, we have defined two structurally different ways of giving various (interleaving, step, pomset, weak history preserving, history preserving) transition system semantics in the context of three event-oriented models — extended prime event structures, stable event structures, and resolvable conflict structures. For each model, we have obtained an isomorphism between the corresponding transition systems for all the semantics except for history preserving one. Also,

we have developed some translations of the event structures from the classes under consideration into resolvable conflict structures and back, so as to compare residual-based transition systems, constructed from the original structures, with the ones constructed from the structures obtained after translation. Further, we have demonstrated that interleaving, step, pomset, weak history preserving bisimulations are captured by the corresponding bisimulations on the transtion systems.

Work on extending our approach (e.g., to precursor [9], probabilistic [29], local [17], dynamic [1] event structures, and to labeled event structures with invisible actions) is presently under way and has yielded promising intermediate results. Another future line of research is to extend our results on comparing two kinds of transition systems to the non-pure case of resolvable conflict structures [14] and to the multiset transition relation.

References

1. Arbach, Y., Karcher, D., Peters, K., Nestmann, U.: Dynamic causality in event structures. *Lecture Notes in Computer Science* **9039** (2015) 83–97.
2. Armas-Cervantes, A., Baldan, B., Garcia-Banuelos, L.: Reduction of event structures under history preserving bisimulation. *J. Log. Algebr. Meth. Program.* 85(6) (2016) 1110-1130.
3. Baier, C., Majster-Cederbaum, M.: The connection between event structure semantics and operational semantics for TCSP. *Acta Informatica* 31 (1994).
4. Baldan, P., Corradini, A., Gadducci, F.: Domains and event structures for fusions. LICS 2017: 1-12
5. Best, E., Gribovskaya, N., Virbitskaite, I.: Configuration- and residual-based transition systems for event structures with asymmetric conflict. Proc. SofSem 2017, *Lecture Notes in Computer Science* **10877** (2018) 117-139.
6. Best, E., Gribovskaya, N., Virbitskaite, I.: From Event-Oriented Models to Transition Systems. Proc. Petri Nets 2018, *Lecture Notes in Computer Science* **10139** (2017) 132–146.
7. Boudol, G.: Flow event structures and flow nets. *Lecture Notes in Computer Science* **469** (1990) 62–95.
8. Crafa, S., Varacca, D., Yoshid, N.: Event structure semantics of parallel extrusion in the pi-calculus. *Lecture Notes in Computer Science* **7213** (2012) 225–239.
9. Fecher, H., Majster-Cederbaum, M.: Event structures for arbitrary disruption. *Fundamenta Informaticae* **68(1-2)** (2005) 103–130.
10. van Glabbeek, R.J.: History preserving process graphs. Report, Stanford University, available at http://boole.stanford.edu/ rvg/pub/abstracts/history.
11. van Glabbeek, R.J.: On the expressiveness of higher dimensional automata. *Theoretical Computer Science* **356(3)** (2006) 265–290.
12. van Glabbeek R.J., Goltz U.: Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica* **37** (2001) 229–327.
13. van Glabbeek R.J., Plotkin G.D.: Configuration structures. Proc. LICS 1995: 199-209
14. van Glabbeek R.J., Plotkin G.D.: Event structures for resolvable conflict. *Lecture Notes in Computer Science* **3153** (2004) 550–561.

15. van Glabbeek R.J., Plotkin G.D.: Configuration structures, event structures and Petri nets. *Theoretical Computer Science* **410(41)** (2009) 4111-4159.
16. van Glabbeek, R.J., Vaandrager F.W.: Bundle event structures and CCSP. *Lecture Notes in Computer Science* **2761** (2003) 57–71.
17. Hoogers, P.W., Kleijn, H.C.M., Thiagarajan, P.S.: An event structure semantics for general Petri nets. *Theoretical Computer Science* **153** (1996) 129–170.
18. Katoen, J.-P.: Quantitative and qualitative extensions of event structures. PhD Thesis, Twente University, 1996.
19. Katoen, J.-P., Langerak, R., Latella, D.: Modeling systems by probabilistic process algebra: an event structures approach. *IFIP Transactions* **C-2** (1993) 253–268.
20. Keller, R.M.: Formal verification of parallel programs. Communications of the ACM, vol 19 nr 7 (1976) 371–384.
21. Langerak, R.: Bundle event structures: a non-interleaving semantics for LOTOS. Formal Description Techniques V. *IFIP Transactions* **C-10** (1993) 331–346.
22. Loogen, R., Goltz, U.: Modelling nondeterministic concurrent processes with event structures. *Fundamenta Informatica* **14(1)** (1991) 39–74.
23. Majster-Cederbaum, M., Roggenbach, M.: Transition systems from event structures revisited. *Information Processing Letters* **67(3)** (1998) 119-124.
24. Nielsen M., Plotkin G., Winskel G.: Petri nets, event structures and domains. *Theoretical Computer Science*, **13**(1) (1981) 85–08.
25. Nielsen M., Thiagarajan P.S.: Regular event structures and finite Petri nets: the conflict-free case. *Lecture Notes in Computer Science* **2360** (2002) 335–351.
26. Winskel G.: Events in computation. PhD Thesis, University of Edinburgh, 1980.
27. Winskel G.: Event structures. *Lecture Notes in Computer Science* **255** (1986) 325–392.
28. Winskel G.: Introduction to event structures. *Lecture Notes in Computer Science* **354** (1989) 364–397.
29. Winskel, G.: Distributed probabilistic and quantum strategies. *Electronic Notes in Theoretical Computer Science* **298** (2013) 403–425.

Compositional Expressiveness of Hybrid Automata

Jafar Akhundov, Michael Reißner, and Matthias Werner

Operating Systems Group, TU Chemnitz, Germany
jafar.akhundov | michael.reissner | matthias.werner@cs.tu-chemnitz.de

Abstract. Linear time-invariant hybrid automata (LTI-HA) have been introduced to model space missions in early design phases. One of LTI-HA's main objectives was therefore to allow a composition of (sub) models. In this paper, we evaluate the expressiveness of the composition in LTI-HA. To compare their expressiveness with the existing hybrid automata formalisms, this work proposes a formal notion of compositional expressiveness. In contrast to the more traditional proofs using simulation relations or bisimilarity to compare single models or their respective behavioral expressiveness or equivalence, compositional expressiveness relies on the complexity of the models and the composition operators enabling the engineer to invest less effort in the modeling process. The following text provides a comparative study of the LTI-HA and several other hybrid formalisms, such as linear hybrid automata and the hybrid I/O automata, with respect to their compositional expressiveness. Specifically, adequacy of their application is discussed based on the case study in space mission feasibility verification.

1 Introduction

Most of the real systems consist of a set of subsystems brought together e.g. by physical aspects or function. A compositional approach to modeling and analysis of such a system relies exclusively on the models of the subsystems, without any holistic information about the composed system, an approach that was first formalized by Gottlob Frege in 1923 [FP93] [Fre05].

For describing space systems during early design phases, a hybrid formalism was proposed in [ATW16] [ASGW16] [ATW15] - Linear Time-Invariant Hybrid Automata (LTI-HA). The adequacy of LTI-HA was demonstrated in [ARW17] by providing an operational semantics for the space mission domain-specific language proposed by Schaus et. al [STF+13]. LTI-HA address the issue of combining continuous dynamics of different discrete states by applying the superposition principle in the composition operator which is extensively applied in the classical control theory as well as the theory of hybrid systems [LA14].

While expressiveness is usually considered from the point of view of behavioral comparison of two formalisms on a meta level, comparison of expressiveness from the compositional point of view lacks the deserved attention [Cas05] [BCH+13] [SA06]. Although many extensions for compositional semantics exist

for hybrid formalisms, only few of them are discussed in terms of expressiveness of models [SY96] [Sif99] [BS00]. Although not desirable, this is an understandable state of affairs since compositional expressiveness of a modeling method is rather a question of taste and usability.

The contribution of the following paper lies in the introduction of the notion of compositional expressiveness and its application for comparison of LTI-HA with other hybrid automata (HA) formalisms, such as linear hybrid automata and hybrid I/O automata. The formal considerations in this work are supported by an use-case based comparison of expressive power of the LTI-HA.

The remaining paper is structured as follows: the next section discusses some of the related work, followed by several supporting definitions in section 3. Section 4 is the core of this paper. First, two motivational examples are provided from the application domain of early spacecraft modeling. Then, several formal metrics are introduced to support the formal definition of the compositional expressiveness which is immediately applied to the LTI-HA formalism with respect to hybrid I/O automata (HIOA) and linear as well as rectangular HA (LHA and RHA, respectively). The paper is finalized with a discussion of behavioral expressiveness of LTI-HA models and some concluding remarks.

2 Related Work

As such, LTI-HA are closely related to the switched, piecewise affine and complementarity systems [LA14]. In [GT04], a class of piecewise affine automata with superposition support is briefly introduced to be immediately applied for modeling biological protein regulatory networks. However, as discussed in [ATW16] and [ATW18], support for invariants and discrete resets could be problematic for the composition. The composition operator has also not been formally introduced for this promising formalism [GT04].

The general framework for timed compositional modeling formalism - timed automata with deadlines where invariants of the classical timed automata are replaced with the notion of deadlines - taken by Sifakis and Yovine [SY96] was extended to hybrid systems in [BS00] by Bornot and Sifakis. They discuss the possible compositional semantics of actions for hybrid systems. It is also assumed that compatible automata operate on disjunct state spaces [BS00].

Van der Shaft and Schumacher [SS01] investigate compositionality from the point of view of dynamic systems and discuss some important properties of the composition operator such as commutativity and associativity.

3 Definitions

The following definitions capture the essential terms necessary for the later discussions [ATW18].

Definition 1 (LTI-HA). *A linear time-invariant hybrid automaton $\mathcal{H}$ is a tuple $(\mathcal{L}, \mathcal{X}, \mathcal{S}_I, \mathcal{S}_O, \mathcal{T}, \mathcal{F})$, where:*

- $\mathcal{L} = \{L_1, \ldots, L_n\}$ *is a set of discrete* ***locations*** *(or* ***modes****);*
- $\mathcal{X}$ *is a set of continuous real-valued variables, called* ***state variables****. Time in this context refers to a designated variable* $t \in \mathcal{X}$*.*
- $\mathcal{S}_I$ *and* $\mathcal{S}_O$ *are two disjoint sets of* ***input and output events****, respectively, which define the event signature of the automaton;*
- $\mathcal{T} \subseteq \mathcal{L} \times 2^{C(\mathcal{X})} \times 2^{\mathcal{S}_I} \times 2^{\mathcal{S}_O} \times \mathcal{L}$ *is a guarded (not necessarily complete)* ***transition relation****, where* $\times$ *denotes a Cartesian product and* $C(\mathcal{X})$ *is a set of all possible constraints over* $\mathcal{X}$*. The* ***guard*** *is thus a triple* $(C, \mathcal{E}, \mathcal{A})$ *with a set of constraints* $C \in 2^{C(\mathcal{X})}$ *a set of input events* $\mathcal{E} \in 2^{S_I}$ *and a set of output events* $\mathcal{A} \in 2^{S_O}$*. The locations along with the transitions (with possible loops) constitute the* ***control graph*** *representing the structure of the given hybrid automaton;*
- *The change of any* $x \in \mathcal{X}$*, except for the time reference* t*, at any time point is described by the* ***flow function*** f_L *of the currently active location describing the continuous change of system state* $x_i(t) = f_{x_i,L}(t)$*. We are restricting the flow functions only to those which are valid solutions for ordinary linear time-invariant differential equations of some order* $k \geq 1$*. This restriction guarantees that flow functions fulfill the superposition property. Therefore, for two simultaneously active locations* L_1 *and* L_2 *of two concurrent hybrid automata the resulting rate of change of a global continuous variable* x_i *is defined as* $x_i(t) = f_{x_i,L_1}(t) + f_{x_i,L_2}(t)$*. Let* $\mathcal{F}$ *denote the set of flow functions for every location in* $\mathcal{L}$ *and* $\dot{t} = 1$ *for all* $L \in \mathcal{L}$*.*

Definition 2 (Initial configuration). $\mathcal{I}$ *is the* ***initial state/configuration*** *of the system* $\sigma(t_0) = (L_\mathcal{I}, V_\mathcal{I})$*, where* $L_\mathcal{I} \in \mathcal{L}$ *is the initial active mode,* $V_\mathcal{I} : \mathcal{X} \mapsto \mathbb{R}$ *is the initial valuation of all the variables in* $\mathcal{X}$ *and* $V_\mathcal{I}(t) = t_0$*.*

Definition 3 (Composition operator). *Any two LTI-HA* $\mathcal{H}^1$ *and* $\mathcal{H}^2$ *for which holds* $\forall x \in \mathcal{X}^1 \cap \mathcal{X}^2 : V_\mathcal{I}^1(x) = V_\mathcal{I}^2(x)$ *are called* compatible*. Given two compatible hybrid automata, the parallel composition* $\mathcal{H}^1 || \mathcal{H}^2$ *produces a new hybrid automaton* $\mathcal{H}^c = (\mathcal{L}^c, \mathcal{X}^c, \mathcal{S}_I^c, \mathcal{S}_O^c, \mathcal{T}^c, F^c)$ *with the initial configuration* $\mathcal{I}^c$*, where the components are defined as follows:*

1. $\mathcal{L}^c = \mathcal{L}^1 \times \mathcal{L}^2 = \{(L_1^1, L_1^2), \ldots, (L_{n_1}^1, L_{n_2}^2)\} =$
   $\{L_{11}^c, L_{12}^c, \ldots, L_{1n_2}^c, \ldots, L_{n_1 n_2}^c\}$
2. $\mathcal{X}^c = \mathcal{X}^1 \cup \mathcal{X}^2$
3. $\mathcal{S}_I^c = (\mathcal{S}_I^1 \backslash \mathcal{S}_O^2) \cup (\mathcal{S}_I^2 \backslash \mathcal{S}_O^1)$
4. $\mathcal{S}_O^c = \mathcal{S}_O^1 \cup \mathcal{S}_O^2$
5. $\forall$ *transitions* $(L_i^x, C^x, \mathcal{E}^x, \mathcal{A}^x, L_k^x), (L_j^y, C^y, \mathcal{E}^y, \mathcal{A}^y, L_l^y)$*:*
   (a) if $\mathcal{E}^x \cap \mathcal{S}_O^y = \emptyset$*:* $\forall L \in \mathcal{L}^y$*:* $\exists((L_i^x, L), C^x, \mathcal{E}^x, \mathcal{A}^x, (L_k^x, L)) \in \mathcal{T}^c$*,*
   (b) $\exists(L_{ij}, C^x \cup C^y, \mathcal{E}^x \cup (\mathcal{E}^y \backslash \mathcal{A}^x), \mathcal{A}^x \cup \mathcal{A}^y, L_{kl}) \in \mathcal{T}^c$*,*
   if $((\mathcal{E}^y \cap \mathcal{A}^x = \mathcal{E}^y \cap \mathcal{S}_O^x)$ *and* $(\mathcal{S}_O^y \cap \mathcal{E}^x = \emptyset))$
   or $((\mathcal{E}^x \cap \mathcal{S}_O^y = \emptyset)$ *and* $(\mathcal{E}^y \cap \mathcal{S}_O^x = \emptyset))$
   where either $x = 1, y = 2$*, or vice versa.*

6. $\forall f_{x_k,L_i}(t) \in \mathcal{F}^1, f_{x_k,L_j}(t) \in \mathcal{F}^2$: $f_{x_k,L_{ij}}(t) = f_{x_k,L_i}(t) + f_{x_k,L_j}(t)$
7. $\mathcal{I}^C = (L^c_{\mathcal{I}_1\mathcal{I}_2}, V^1_{\mathcal{I}} \cup V^2_{\mathcal{I}})$

The operator is not defined for not compatible LTI-HA.

4 Compositional Expressiveness

The following section provides two motivational examples: one informally discusses a use case when set-based directional event semantics is more expressive than any semantics based on a singleton labels, and the second shows how superposition principle allows a modeling engineer to apply the divide and conquer approach more effectively and therefore reduces modeling effort.

4.1 Example 1

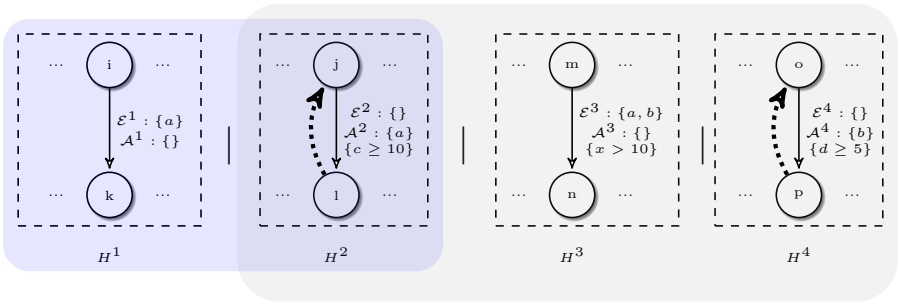


Fig. 1. Expressiveness of the LTI-HA set-based directional event semantics

Consider the interaction patterns between the four transitions in the four LTI-HA depicted in Figure 1. The automata are representing, from left to right: a downlink module of a satellite moving on a Low Earth Orbit(LEO), primary ground station A, synchronization module and the secondary ground station B. Without loss of generality, we are only considering the parts of the automata representing interactions with one another and not the specifics of their dynamics. Synchronization events a and b represent availability of the ground stations A and B, respectively. Synchronization module is responsible for synchronizing the clocks of the satellite, a procedure carried out only when both ground stations are available. This is done to eliminate possible synchronization faults (see, for example, [DHSS95] or [LMS86]).

It is clear that transitions $j \to l$ and $o \to p$ can be taken independently, while the transitions $i \to k$ and $m \to n$ cannot. The transition $m \to n$ can only be taken simultaneously with the both enabled transitions $j \to l$ and $o \to p$. The transition $i \to k$ will only be synchronized with the transition $j \to l$. The paths of non-zero length from l to j and p to o are represented with thick dotted arrows. If the locations j and o can simultaneously be active with valuations $\{c \geq 10, d < 5\}$ and $\{c \geq 10, d \geq 5\}$, then, obviously, transitions $j \to l$ and $o \to p$ are not always taken synchronously.

For HIOA or structured HIOA (SHIOA [MLL06] [Mit07]) to model this system, a relabling procedure consisting of two modifications may be performed: first, both ground stations have to be modeled as a single automaton; second, the transition with two events being simultaneous, a new event has to be introduced, $e = \{a, b\}$ which would synchronize with the $m \to n$ transition. The same holds for the timed automata if the continous dynamics could be modeled with only clock variables [BY04].

Set-based undirectional hybrid automata formalism like [RTP96] may be suitable for modeling the system described in Figure 1. However, the composition operator in [RTP96] only considers synchronous actions in case when there is a non-empty intersection between them. Let us assume that the transitions $i \to k$, $m \to n$ and $o \to p$ are labeled with $\Sigma_{i\to k} = \{a\}$, $\Sigma_{m\to n} = \{a, b\}$ and $\Sigma_{o\to p} = \{b\}$, respectively. The above interaction is indeed preserved. However, if we first compose $\mathcal{H}^1$ with $\mathcal{H}^4$ and only then with $\mathcal{H}^3$, the information about possible simultaneity of events a and b is lost. Since composition operator in [RTP96, p.7] only considers the labeled dependencies between transitions pairwise, transition with $\Sigma_{m\to n} = \{a, b\}$ will never be enabled. However, merging $m \to n$ first with either $i \to k$ or $o \to p$ solves the problem. Hence, composition operator in [RTP96, p.7] is not associative. Our composition operation is both associative and supports event directions making them more intuitional.

Obviously, not set-based undirectional based are less expressive than the last two discussed methods. This clearly demonstrates the advantage of the directional set-based event synchronization: directionality makes events more intuitional while assigning more than one label to a transition, more complex communication patterns are possible.

4.2 Example 2

As a second example, consider the two automata from [ATW18] in Figure 2.

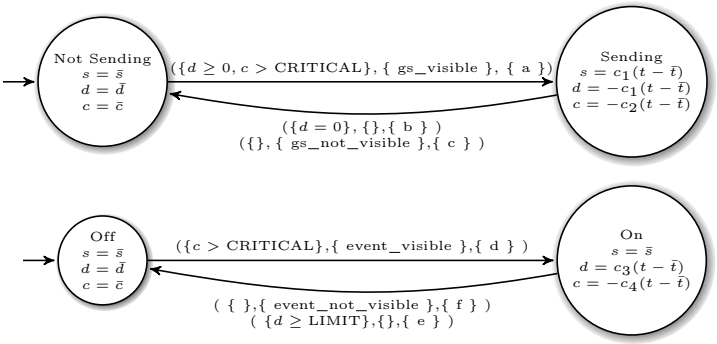


Fig. 2. Downlink Satellite Component and Experimental Payload (Camera)

In the early conceptual phase of development, the satellite downlink module which sends gathered information back to Earth is modeled as having only two distinct states: *Sending*, when a ground station is visible and there is data to send, or *Not Sending*, when either no ground station is available or there is no

data to be sent (or both). In the *Sending* state the rate of change of available data and sent data is the same with opposite signs, whereas in the *Not Sending* state both parameters remain constant.

The valuable data which is sent back to Earth comes from a camera which can rotate and fix on the events of interest. Hence, depending on whether an event is visible or not, the camera can also be in two states: On and Off. The data can only be written if there is enough free storage, therefore the constant *LIMIT* represents the maximum data that can be stored at any instance. The constant *CRITICAL* stands for the lowest level of battery charge needed for any of the components to start.

The continuous variables s, d, c stand for the amount of data sent, the data still stored on the satellites storage unit and the battery charge, respectively. c_1-c_4 are some predefined constants. The downlink component can only be activated when a ground station is visible which is represented by the *gs_visible* event.

Since LTI-HA explicitly support the superposition principle for flow functions, output trajectories do not need to be exclusive. This allows to combine effects of both automata on the same continuous variable, in this case all three of them: s, d, c by the composition operator [ATW18].

(S)HIOA explicitly forbid superposition of the output trajectories [LSV03, p.131,p.141] [Mit07, p.35]. That is, for any formalism without support for superposition to model a system where some of the components have overlapping continuous output trajectories, it is necessary to model all of those components as a single automaton with the cartesian product of all of the corresponding locations.

Other hybrid automata formalisms follow either the same strategy as (S)HIOA or imply an agreement of flow conditions between the locations [ATW18]. Thus, they would also require an engineer to explicitly specify all of the possible combinations of flows thus eliminating the advantages of compositional analysis altogether.

4.3 Generalization and Metrics

Transitions Given m LTI hybrid automata $\mathcal{H}^1, ..., \mathcal{H}^m$, we define a synchronization function sync: $\mathcal{T}_\cup \mapsto 2^{\mathcal{T}_\cup}$ that maps the transition τ^i to the set of all transitions which synchronize with it:

$$\text{sync}(\tau^i) = \begin{cases} \{\tau^j | \sigma_{\tau^i} = \sigma_{\tau^j}, i \neq j\} & \text{for non-directional event semantics (LHA [ACHH93] [NOSY93] [Hen00], RHA [Ras05] [PV94] [Hen00], HA [Áb12])} \\ \{\tau^j | \Sigma_{\tau^i} \cap L_j = \Sigma_{\tau^j} \cap L_i \neq \emptyset, i \neq j\} & \text{for set-based non-directional event semantics (LHA [RTP96])} \\ \{\tau^j | \sigma_{a,\tau^i} = \sigma_{e,\tau^j}, i \neq j\} & \text{for directional event semantics (HIOA [LSV03], TA [BY04])} \\ \{\tau^j | \mathcal{A}_{\tau^i} \cap \mathcal{E}_{\tau^j} = \mathcal{S}^i_O \cap \mathcal{E}_{\tau^j} \neq \emptyset, i \neq j\} & \text{for set-based event semantics (LTI-HA [ATW18])} \end{cases}$$

where $\mathcal{T}_\cup = \bigcup_{i=1..m} \mathcal{T}^i$ with $\mathcal{T}^i$ from $\mathcal{H}^i$, σ_{τ^i} is the label on the transition τ^i in the automaton $\mathcal{H}^i$, $\sigma_{e,\tau^i}, \sigma_{a,\tau^i}$ are the input and output labels on the

transition τ in the automaton $\mathcal{H}^i$. Σ_{τ^i} is a set of labels assigned to transition τ^i respectively. L_i is the set of labels of the LHA i [RTP96]. τ^i is any transition out of $\mathcal{T}_\cup$. Without loss of generality, we assume that for the directional event semantics, there is always only a single automaton generating every consumable event [ATW18] [LSV03].

The function $rsync : \mathcal{T}_\cup \mapsto 2^{\mathcal{T}_\cup}$ maps the set of all transitions τ^i to the set of all transitions which generate the necessary inputs for the transition at hand:

$$\text{rsync}(\tau^i) = \begin{cases} \text{same as sync}(\tau^i) & \text{for non-directional event semantics} \\ & \text{(LHA [ACHH93] [NOSY93] [Hen00],} \\ & \text{RHA [Ras05] [PV94] [Hen00], HA [Áb12])} \\ \text{same as sync}(\tau^i) & \text{for set-based non-directional event semantics} \\ & \text{(LHA [RTP96])} \\ \{\tau^j | \sigma_{e,\tau^i} = \sigma_{a,\tau^j}, i \neq j\} & \text{for directional event semantics} \\ & \text{(HIOA, TA)} \\ \{\tau^j | \mathcal{E}_{\tau^i} \cap \mathcal{A}_{\tau^j} = \mathcal{E}_{\tau^i} \cap \mathcal{S}_O^j \neq \emptyset, i \neq j\} & \text{for set-based event semantics} \\ & \text{(LTI-HA)} \end{cases}$$

Therefore, $rsync$ defines a different set of values in the cases of directional event semantics. Table 1 provides a comparative study of compositional properties defined below for HA depending on the types of event semantics: directional or non-directional, set-based or singular (singleton).

$|sync(\tau^i)|$ provides the number of transitions from other automata with whom the given transition may synchronize. For the directed event semantics, output events are usually observable by any other automata [LSV03] [ATW18]. Obviously, for all the formalisms, this number is greater equal than zero.

The $|\text{rsync}(\tau^i)|$-row represents the number of transitions having as outputs the inputs of τ^i. Since the inverse function is defined in the same way as the original for the undirectional event semantics, no change is observed here.

$\forall j : |\{\tau^j | \tau^j \in \text{rsync}(\tau^i), \mathcal{E}_{\tau^j} \cap \mathcal{A}_{\tau^j} \subset \mathcal{E}_{\tau^j} \cap \mathcal{H}^i\}|$ is the number of transitions which possibly generate the necessary input for the given transition but do not have all the events generated by the automaton containing them. This is a critical condition to be fulfilled during composition, if there is a single producer for each event. It is guaranteed by the condition 5b for the composition operator of LTI-HA and the condition (3) for the event synchronization in [RTP96, p.7].

$|\{s | s \subset \text{rsync}(\tau^i), \forall \tau^j, \tau^k \in s : j = k, \nexists s' \subset \text{rsync}(\tau^i)$ with $\tau^l \in s', \exists \tau_j \in s : j = l\}|$ is the number of hybrid automata generating the necessary events for a given transition. In the case of singular directional event semantics it is only possible to synchronize with a single producer of an event for the HIOA [LSV03] [Mit07]. Timed automata in the UPPAAL [BLP+96] allow for several generators of a single event. This row is emphasized since this is a case of an increased expressiveness of set-based labeling formalisms in contrast to singular labeling mechanisms.

$\max(|s | s \subset \text{rsync}(\tau^i), \forall \tau^j, \tau^k \in s : j = k, \nexists s' \subset \text{rsync}(\tau^i)$ with $\tau^l \in s', \exists \tau_j \in s : j = l\}|)$ represents the maximum number of transitions generating the necessary events for the given transition, per automaton. For all the cases, the number of those transitions can be non-zero, however, for the cases of set-based labelings,

the events should be exactly those which lie in the intersection set of the whole automaton and the given transition τ^i, since synchronization (and structural merging during composition) only occur pairwise in the existing formalisms.

The following row is another case where set-based approaches have a definite advantage with respect to singular cases: it represents the number of possible transitions from other automata which can consume *subsets* of the given event label. Obviously, no such thing exists for singular cases, and only those having strictly the same labeling will be able to sychronize.

Event / Semantics / Properties	Non-directional		Directional	
	Singular	Set-based	Singular	Set-based
Examples	[ACHH93] [NOSY93] [Hen00]	[RTP96]	[LSV03] [Mit07] [BY04]	[ATW18]
$\lvert sync(\tau^i)\rvert$	≥ 0	≥ 0	≥ 0	≥ 0
$\lvert \text{rsync}(\tau^i)\rvert$	> 0	≥ 0	> 0	≥ 0
$\forall j: \lvert\{\tau^j \lvert \tau^j \in \text{rsync}(\tau^i), \mathcal{E}_{\tau^i} \cap \mathcal{A}_{\tau^j} \subset \mathcal{E}_{\tau^i} \cap \mathcal{S}_O^j\}\rvert$	-	0	-	0
$\lvert\{s \lvert s \subset \text{rsync}(\tau^i), \forall \tau^j, \tau^k \in s : j = k, \nexists s' \subset \text{rsync}(\tau^i) \text{ with } \tau^l \in s', \exists \tau_j \in s : j = l\}\rvert$	0 or 1	≥ 0	0 or 1 (> 0 [BY04])	≥ 0
$\max(\lvert s \lvert s \subset \text{rsync}(\tau^i), \forall \tau^j, \tau^k \in s : j = k, \nexists s' \subset \text{rsync}(\tau^i) \text{ with } \tau^l \in s', \exists \tau_j \in s : j = l\}\rvert)$	≥ 0	≥ 0	≥ 0	≥ 0
$\forall A^i$ from $\tau^i \in \mathcal{T}_\cup, \lvert\mathcal{A}^i\rvert > 1$: $\lvert\tau^j$ from $\mathcal{H}^j$, $j \neq i$ with $s = \mathcal{E}^j \cap \mathcal{A}^i \subset \mathcal{A}^i, s \neq \emptyset\rvert$	0	≥ 0	0	≥ 0
Circular event dependencies	no	no	yes	yes
Stutter transitions	yes	yes	no	$\mathcal{E}^i = \emptyset, \mathcal{A}^i = \emptyset, \mathcal{C}^i = \emptyset$

Table 1. Event semantics properties (B $\triangleq$ Blocking, NB $\triangleq$ Non-blocking)

Before we discuss the next row of the table, the following definitions are due.

Definition 4 (Event Dependency of Guards). *Two guards - $g_1 = (C_1, \mathcal{E}_1, \mathcal{A}_1)$ from $\mathcal{H}^1, g_2 = (C_2, \mathcal{E}_2, \mathcal{A}_2)$ from $\mathcal{H}^2$ - are said to be **event-dependent** iff $\mathcal{E}_1 \cap \mathcal{A}_2 \neq \emptyset \vee \mathcal{E}_2 \cap \mathcal{A}_1 \neq \emptyset$. The guard g_1 is said to be **event-dependent** on the second automaton iff $\mathcal{E}_1 \cap \mathcal{S}_O^2 \neq \emptyset$.*

Event (in)dependency of guards consequently implies event-(in)dependency of the corresponding transitions. It is clear from the constraint $\mathcal{S}_I \cap \mathcal{S}_O = \emptyset$ that those two guards cannot be in the same automaton.

Definition 5 (Cycle of Event Dependencies for a Set of Guards). *A set of guards G of size $k \geq 2$ has a cycle of event dependencies if there is a sequence of these guards $(g_0, ..., g_{k-1})$, such that $\forall i, 0 \leq i < k : \mathcal{A}_i \cap \mathcal{E}_{(i+1) mod\ k} \neq \emptyset$.*

Clearly, circular event dependencies are only possible for the directional cases. The last row shows wether the formalism has an explicit support for stutter transitions. HIOA have eliminated them in the later redefinitions to avoid compositionality issues [LSV03]. LTI-HA do not have implicit stutter transitions. However, they can be explicitly defined, as provided in the Table 1. These transitions, however, are always enabled and will be taken immediately when their

start-location is entered. They also pose a danger of introducing time locks (time convergence), should there be a cycle containing only stutter transitions.

The relabling procedure discussed in the first example of this section, can be formalized as follows:

$$\forall \tau^i \text{ from } \mathcal{H}^i \text{ where } |\mathcal{E}_{\tau^i}| > 1 : \ \forall \mathcal{H}^j \text{ with } \mathcal{S}_O^j \cap \mathcal{E}_{\tau^i} \neq \emptyset : \ \Big|\Big|_{\mathcal{H}^j} \tag{1}$$

The labeled event set $\mathcal{E}_{\tau^i}$ will be then renamed to a single letter in the composed automaton. Although we do not account here for the cascading dependencies of the events, it should be clear that in the worst case all of the automata $\mathcal{H}^j,\ j = 1..m$ will have to be combined with each other by the user. E.g. consider the scenario where one LTI-HA consumes all of the events generated by other LTI-HA over a single transition. Furthermore, for any set of event labels $\mathcal{A}$, all of the possible subset of those events can be consumed by a different automaton. That is, for every possible set from $2^{\mathcal{A}}$, a new transition has to be introduced in the automaton containing a transition labeled with $\mathcal{A}$.

Locations As discussed in the second example of this section, the modeling engineer has to account for all of the possible combinations of continuous flows which could become exponential in the minimum number of states of the HA, if for all $|\mathcal{L}^i|$ holds $|\mathcal{L}^i| \geq 2$. That is, for 10 LTI-HA with three states per automaton, 3^{10} locations would be required to model in any formalism not supporting superposition: i.e. (S)HIOA, LHA, RHA, HA, etc.

Formally, in the worst case for a set of m LTI hybrid automata $\mathcal{H}^1, ..., \mathcal{H}^m$:

$$\text{if } \exists X \subseteq \mathcal{X}_U, \mathcal{X}_U = \bigcup_{i=1..m} \mathcal{X}^i \text{ where } \forall x \in X, \forall L_j^i : \ \dot{x}_{L_j^i} \neq 0 \tag{2}$$
$$\text{then } \mathbb{L} = \bigtimes_{i=1}^{m} \mathcal{L}^i, \ |\mathbb{L}| = \prod_{i=1}^{m} |\mathcal{L}^i|$$

where $\mathbb{L}$ is the total set of the composed locations.

Definition of Compositional Expressiveness Considering the above metrics, we can now define the notion of compositional expressiveness for hybrid automata. It amounts to the information a modeling engineer uses to describe the system components before the composed system is built. Clearly, since the composition operator has exponential run-time and produces a structure with an exponential number of nodes, it is desirable to shift the modeling complexity into the "pre-composition phase" since the modeling effort is then linear in the number of nodes and edges.

We rely on the number of edges ($\mathcal{T}$) as well as the number of vertices ($\mathcal{L}$) needed to describe a system S. We assume that for both formalisms under comparison, a minimal representation can be achieved, and that there is some behavioral equivalence satisfied for models of S in the respective formalisms (e.g. bisimilarity).

Definition 6 (Compositional Expressiveness). *A modeling formalism A is compositionally more expressive than a modeling formalism B ($A \ni B$) if, for describing a system S (or a family of systems):*

$$1.\ \mathcal{T}_A(S) < \mathcal{T}_B(S)$$
$$2.\ \mathcal{L}_A(S) < \mathcal{L}_B(S)$$

for a minimal representations of S in A, $\mathcal{H}_A(S)$, and B, $\mathcal{H}_B(S)$, respectively.

Proposition 1. *For a family of systems where the the condition from the equation (2) holds or $\exists \tau^i$ from $\mathcal{T}_\cup$ with $|\mathcal{E}_{\tau^i}| > 1$, linear time-invariant hybrid automata are compositionally more expressive than HIOA.*

Proof. The proof follows from the discussions and metrics from the this section.

5 Discussion

This paper has introduced an initial approach at formalizing the notion of compositional expressiveness in terms of events labeling and continuous flow combinations. It has been directly applied to demonstrate the superior compositional expressiveness of LTI-HA with respect to HIOA.

Timing of transitions was never considered in this work, as well as the modal extensions of guards [SY96, p.5]. In the extended version of this work, we plan to include timeliness of events as well as the modalities for combining the guards which are assigned to the transitions into the notion of compositional expressiveness. We also intend to adopt the notion of HA comparability of Lynch et. al. [LSV03] and put the LTI-HA in the context of other formalisms in terms of behavioral expressiveness.

Associativity is an important property for composition of discrete event systems in general and hybrid automata in particular [SS01] [CL10]. Proving it for LTI-HA is not trivial due to the complex event semantics and remains as a future challenge.

The relabling procedure discussed in section 4 can potentially be extended to a full algorithm which would transform a system of LTI-HA into a system of HIOA which is a powerful modeling method with tool support [Fre05]. This algorithm would also require a behavioral expressiveness relation established between the two formalisms which we also intend as a future work. A formal meta-language that could be used to define simulation relations could be the extension theorem [BS00] [SY96] or timed transition systems [Cas05] [BCH$^+$13].

References

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1993.

[ARW17] Jafar Akhundov, Michael Reißner, and Matthias Werner. Using Hybrid Automata for Early Spacecraft Design Evaluation. In *Proceedings of the 26th International Workshop on Concurrency, Specification and Programming*, Warsaw, Poland, September 2017.

[ASGW16] Jafar Akhundov, Volker Schaus, Andreas Gerndt, and Matthias Werner. Using Timed Automata to Check Space Mission Feasibility in the Early Design Phases. In *IEEE Aerospace 2016 Proceedings*, Big Sky, Montana, USA, March 2016.

[ATW15] Jafar Akhundov, Peter Tröger, and Matthias Werner. Considering Concurrency in Early Spacecraft Design Studies. In *CS&P 2015 Proceedings*, pages 22–30, Rzeszow, Poland, 2015.

[ATW16] Jafar Akhundov, Peter Tröger, and Matthias Werner. Superposition Principle in the Composable Hybrid Automata. In *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming*, pages 125–140, Rostock, Germany, September 2016.

[ATW18] Jafar Akhundov, Peter Tröger, and Matthias Werner. Superposition Principle in Composable Hybrid Automata. *Fundamenta Informaticae*, 157(Concurrency, Specification, and Programming: Special Issue of Selected Papers of CS&P 2016):321–339, 2018.

[BCH+13] B. Bérard, F. Cassez, S. Haddad, D. Lime, and O.H. Roux. The expressive power of time Petri nets. *Theoretical Computer Science*, 474:1–20, February 2013.

[BLP+96] Johan Bengtsson, Fredrik Larsson, Paul Pettersson, Wang Yi, Palle Christensen, Jesper Jensen, Per Jensen, Kim Larsen, and Thomas Sorensen. *UPPAAL: a Tool Suite for Validation and Verification of Real-Time Systems*. 1996.

[BS00] Sébastien Bornot and Joseph Sifakis. On the Composition of Hybrid Systems. In M. Kemal Inan and Robert P. Kurshan, editors, *Verification of Digital and Hybrid Systems*, pages 293–322. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[BY04] Johan Bengtsson and Wang Yi. Timed Automata: Semantics, Algorithms and Tools. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, pages 87–124. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[Cas05] B. Bérard F. Cassez. Comparison of the expressiveness of timed automata and time Petri nets. In *In Proc. FORMATS'05, vol. 3829 of LNCS*, pages 211–225. Springer, 2005.

[CL10] Christos G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*. Springer Publishing Company, Incorporated, 2nd edition, 2010.

[DHSS95] Danny Dolev, Joseph Y. Halpern, Barbara Simons, and Ray Strong. Dynamic fault-tolerant clock synchronization. *J. ACM*, 42(1):143–185, January 1995.

[FP93] G. Frege and G. Patzig. *Logische Untersuchungen*. Kleine Reihe Vandenhoeck und Ruprecht. Vandenhoeck & Ruprecht, 1993.

[Fre05] Goran Fedja Frehse. *Compositional verification of hybrid systems using simulation relations*. [Sl: sn], 2005.

[GT04] R. Ghosh and C. Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch protein signalling. *IEE Proceedings - Systems Biology*, 1(1):170–183, June 2004.

[Hen00] ThomasA. Henzinger. The Theory of Hybrid Automata. In M.Kemal Inan and RobertP. Kurshan, editors, *Verification of Digital and Hybrid Systems*, volume 170 of *NATO ASI Series*, pages 265–292. Springer Berlin Heidelberg, 2000.

[LA14] Hai Lin and Panos J. Antsaklis. Hybrid Dynamical Systems: An Introduction to Control and Verification. *Foundations and Trends® in Systems and Control*, 1(1):1–172, 2014.

[LMS86] Leslie Lamport and P. M. Melliar-Smith. Byzantine Clock Synchronization. *Operating Systems Review*, 20(3):10–16, 1986.

[LSV03] Nancy Lynch, Roberto Segala, and Frits Vaandrager. Hybrid I/O Automata. *Inf. Comput.*, 185(1):105–157, August 2003.

[Mit07] Sayan Mitra. *A Verification Framework for Hybrid Systems*. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA 02139, September 2007.

[MLL06] Sayan Mitra, Daniel Liberzon, and Nancy Lynch. Verifying Average Dwell Time by Solving Optimization Problems. In Ashish Tiwari and Joao P. Hespanha, editors, *Hybrid Systems: Computation and Control (HSCC 06)*, LNCS, Santa Barbara, CA, March 2006. Springer.

[NOSY93] Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. An approach to the description and analysis of hybrid systems. *Hybrid Systems*, pages 149–178, 1993.

[PV94] Anuj Puri and Pravin Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In David L. Dill, editor, *Computer Aided Verification: 6th International Conference, CAV '94 Stanford, California, USA, June 21–23, 1994 Proceedings*, pages 95–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[Ras05] Jean-François Raskin. An Introduction to Hybrid Automata. In Dimitrios Hristu-Varsakelis and William S. Levine, editors, *Handbook of Networked and Embedded Control Systems*, pages 491–517. Birkhäuser Boston, Boston, MA, 2005.

[RTP96] R. Alur, T. A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, March 1996.

[SA06] S. Di Cairano and A. Bemporad. An Equivalence Result between Linear Hybrid Automata and Piecewise Affine Systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 2631–2636, December 2006.

[Sif99] Joseph Sifakis. The Compositional Specification of Timed Systems - A Tutorial. In *Proceedings of the 11th International Conference on Computer Aided Verification*, CAV '99, pages 2–7, London, UK, UK, 1999. Springer-Verlag.

[SS01] A.J. van der Schaft and J. M. Schumacher. Compositionality issues in discrete, continuous, and hybrid systems. *International Journal of Robust and Nonlinear Control*, 2001.

[STF+13] Volker Schaus, Michael Tiede, Philipp M. Fischer, Daniel Lüdtke, and Andreas Gerndt. A Continuous Verification Process in Concurrent Engineering. In *AIAA Space Conference*, September 2013.

[SY96] Joseph Sifakis and Sergio Yovine. Compositional Specification of Timed Systems (Extended Abstract). In *STACS*, 1996.

[Áb12] Erika Ábrahám. *Modeling and Analysis of Hybrid Systems: Lecture Notes*. RWTH Aachen University, April 2012.

The Influence of the Test Operator on the Expressive Powers of PDL-Like Logics

Linh Anh Nguyen

Institute of Informatics, University of Warsaw,
Banacha 2, 02-097 Warsaw, Poland
nguyen@mimuw.edu.pl

Abstract. Berman and Paterson proved that Test-Free PDL is weaker than PDL. As the description logics $\mathcal{ALC}_{trans}$ and $\mathcal{ALC}_{reg}$ are, respectively, variants of Test-Free PDL and PDL, there is a concept of $\mathcal{ALC}_{reg}$ that is not equivalent to any concept of $\mathcal{ALC}_{trans}$. Generalizing this, we show that there is a concept of $\mathcal{ALC}_{reg}$ that is not equivalent to any concept of the logic that extends $\mathcal{ALC}_{trans}$ with inverse roles, nominals, qualified number restrictions, the universal role and local reflexivity of roles. We also provide some results for the case with RBoxes and TBoxes. One of them states that tests can be eliminated from TBoxes of the deterministic Horn fragment of $\mathcal{ALC}_{reg}$.

1 Introduction

Propositional Dynamic Logic (PDL) is a well-known modal logic for reasoning about computer programs [5,7]. Its variant $\mathcal{ALC}_{reg}$ is a description logic (DL) for reasoning about terminological knowledge [16]. Berman and Paterson [2] proved that Test-Free PDL is weaker than PDL. In particular, they gave a formula of PDL that is not equivalent to any formula of Test-Free PDL. This means that there is a concept of $\mathcal{ALC}_{reg}$ that is not equivalent to any concept of $\mathcal{ALC}_{trans}$ (a variant of Test-Free PDL). While bisimulations are usually used for separating the expressive powers of modal and description logics (see, e.g., [3,4,10]), the proof given by Berman and Paterson [2] exploits the fact that "over a single symbol alphabet, the regular sets are precisely those which are ultimately periodic" (see [6, Theorem 3.1.2]) and is somehow similar to the proof of that connectivity is inexpressible in first-order logic.

Generalizing the result and method of Berman and Paterson, in Section 3 we prove that there is a concept of $\mathcal{ALC}_{reg}$ that is not equivalent to any concept of the DL $\mathcal{ALCIOQUS}elf_{trans}$, which extends $\mathcal{ALC}_{trans}$ with inverse roles ($\mathcal{I}$), nominals ($\mathcal{O}$), qualified number restrictions ($\mathcal{Q}$), the universal role ($\mathcal{U}$) and local reflexivity of roles ($\mathcal{S}elf$) as of the DL $\mathcal{SROIQ}$ [8]. That is, extending $\mathcal{ALC}_{trans}$ with the features $\mathcal{I}$, $\mathcal{O}$, $\mathcal{Q}$, $\mathcal{U}$ and $\mathcal{S}elf$ does not help in expressing the test operator. Modifying the proof of Berman and Paterson [2] for dealing with the features $\mathcal{O}$, $\mathcal{Q}$, $\mathcal{U}$ and $\mathcal{S}elf$ can be done in a rather straightforward way (see our Lemmas 1, 3, 4 and their proofs). However, dealing with inverse roles ($\mathcal{I}$)

requires an advanced refinement, as regular sets over an alphabet consisting of an atomic role and its inverse need not be ultimately periodic. The proof of our Lemma 2 is more sophisticated than the proof of [6, Theorem 3.1.2].

In Section 4, we provide a result stating that using regular RBoxes and acyclic TBoxes for $\mathcal{ALCIOQUSelf}_{trans}$ does not help in expressing tests, but using simple stratified TBoxes under the stratified semantics on the background allows us to express every concept by another without tests. A further result states that tests can be eliminated from TBoxes of the deterministic Horn fragment of $\mathcal{ALC}_{reg}$. This suggests that tests can be eliminated from tractable[1] Horn fragments of PDL-like logics.

2 Preliminaries

This section provides notions and definitions related with syntax and semantics of DLs [1]. We denote the sets of concept names, role names and individual names by $\mathbf{C}$, $\mathbf{R}_+$ and $\mathbf{I}$, respectively. A concept name is an *atomic concept*, a role name is an *atomic role*. Let $\mathbf{R} = \mathbf{R}_+ \cup \mathbf{R}_-$, where $\mathbf{R}_- = \{\overline{r} \mid r \in \mathbf{R}_+\}$ and $\overline{r}$ is called the *inverse* of r. We call elements of $\mathbf{R}$ *basic roles*. We distinguish a subset of $\mathbf{R}_+$ whose elements are called *simple roles*. If $r \in \mathbf{R}_+$ is a simple role, then $\overline{r}$ is also a simple role. The set $\Sigma = \mathbf{C} \cup \mathbf{R}_+ \cup \mathbf{I}$ is called the *signature*.

Let $\Phi \subseteq \{\mathcal{I}, \mathcal{O}, \mathcal{Q}, \mathcal{U}, \mathcal{Self}\}$, where the symbols mean inverse roles, nominals, qualified number restrictions, the universal role and local reflexivity of roles, respectively. Roles and concepts of the DLs $\mathcal{ALC}$, $\mathcal{ALC}{+}\Phi$, $(\mathcal{ALC}{+}\Phi)_{trans}$ and $(\mathcal{ALC}{+}\Phi)_{reg}$ are defined as follows.

If $\mathcal{L} = \mathcal{ALC}$, then:

- if $r \in \mathbf{R}_+$, then r is a role of $\mathcal{L}$,
- if $A \in \mathbf{C}$, then A is a concept of $\mathcal{L}$,
- $\top$ and $\bot$ are concepts of $\mathcal{L}$,
- if C and D are concepts of $\mathcal{L}$ and R is a role of $\mathcal{L}$, then $\neg C$, $C \sqcup D$, $C \sqcap D$, $\exists R.C$ and $\forall R.C$ are concepts of $\mathcal{L}$.

If $\mathcal{L} = \mathcal{ALC}{+}\Phi$, then additionally:

- if $\mathcal{I} \in \Phi$ and R is a role of $\mathcal{L}$, then $\overline{R}$ is a role of $\mathcal{L}$,
- if $\mathcal{O} \in \Phi$ and $a \in \mathbf{I}$, then $\{a\}$ is a concept of $\mathcal{L}$,
- if $\mathcal{Q} \in \Phi$, $n \in \mathbb{N}$, C is a concept of $\mathcal{L}$, R is a simple role of $\mathcal{L}$ (i.e., a simple role that is a role of $\mathcal{L}$), then $\geq n\, R.C$ and $\leq n\, R.C$ are concepts of $\mathcal{L}$,
- if $U \in \Phi$, then U is a role of $\mathcal{L}$,
- if $\mathcal{Self} \in \Phi$ and $r \in \mathbf{R}_+$, then $\exists r.\mathcal{Self}$ is a concept of $\mathcal{L}$.

If $\mathcal{L} = (\mathcal{ALC}{+}\Phi)_{trans}$, then additionally:

- ε is a role of $\mathcal{L}$,
- if R and S are roles of $\mathcal{L}$ and are different from U, then $R \sqcup S$, $R \circ S$ and R^* are roles of $\mathcal{L}$.

[1] I.e., with a PTIME or lower data complexity.

$\bot^{\mathcal{I}} = \emptyset \quad \top^{\mathcal{I}} = \Delta^{\mathcal{I}} \quad \{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\} \quad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$\overline{R}^{\mathcal{I}} = (R^{\mathcal{I}})^{-1}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\varepsilon^{\mathcal{I}} = \{\langle x, x\rangle \mid x \in \Delta^{\mathcal{I}}\}$
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists y\,(\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$	$(R \circ S)^{\mathcal{I}} = R^{\mathcal{I}} \circ S^{\mathcal{I}}$
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \forall y\,(\langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}})\}$	$(R \sqcup S)^{\mathcal{I}} = R^{\mathcal{I}} \cup S^{\mathcal{I}}$
$(\exists R.\mathcal{S}elf)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \langle x, x\rangle \in R^{\mathcal{I}}\}$	$(R^*)^{\mathcal{I}} = (R^{\mathcal{I}})^*$
$(\geq n\, R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$	$(C?)^{\mathcal{I}} = \{\langle x, x\rangle \mid x \in C^{\mathcal{I}}\}$
$(\leq n\, R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$	$U^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Fig. 1. Semantics of complex concepts and complex roles.

If $\mathcal{L} = (\mathcal{ALC}{+}\Phi)_{reg}$, then additionally:

- if C is a concept of $\mathcal{L}$, then $C?$ is a role of $\mathcal{L}$.
  This constructor is called the *test operator*.

When $\Phi = \emptyset$, we shorten the names $(\mathcal{ALC}{+}\Phi)_{trans}$ and $(\mathcal{ALC}{+}\Phi)_{reg}$ to $\mathcal{ALC}_{trans}$ and $\mathcal{ALC}_{reg}$, respectively. Similarly, we write $\mathcal{ALCIOQUS}elf_{trans}$ to denote $(\mathcal{ALC}{+}\Phi)_{trans}$ with $\Phi = \{\mathcal{I}, \mathcal{O}, \mathcal{Q}, \mathcal{U}, \mathcal{S}elf\}$, and so on.

We denote atomic concepts by letters like A or B, atomic roles by letters like r or s, and individual names by letters like a or b. We use letters C and D to denote (arbitrary) concepts, R and S to denote (arbitrary) roles.

An *interpretation* is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set, called the *domain*, and $\cdot^{\mathcal{I}}$ is the *interpretation function* of $\mathcal{I}$ that maps each $a \in \mathbf{I}$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each $A \in \mathbf{C}$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and each $r \in \mathbf{R}_+$ to a relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is extended to interpret complex roles and concepts as specified in Figure 1.

Concepts C and D are *equivalent*, denoted by $C \equiv D$, if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$. Similarly, roles R and S are *equivalent*, denoted by $R \equiv S$, if $R^{\mathcal{I}} = S^{\mathcal{I}}$ for all interpretations $\mathcal{I}$.

If $\mathcal{L}$ is a sublogic of $\mathcal{L}'$ (like $(\mathcal{ALC}{+}\Phi)_{trans}$ is a sublogic of $(\mathcal{ALC}{+}\Phi)_{reg}$), then we say that $\mathcal{L}$ is *weaker* (or *less expressive*) than $\mathcal{L}'$ (in expressing concepts) if there exists a concept C of $\mathcal{L}'$ that is not equivalent to any concept of $\mathcal{L}$.

2.1 RBoxes

A finite set $\mathcal{S}$ of context-free production rules over $\mathbf{R}$ is called a *context-free semi-Thue system* over $\mathbf{R}$. It is *symmetric* if $\overline{R} \to \overline{S}_k \ldots \overline{S}_1$ belongs to $\mathcal{S}$ for every production rule $R \to S_1 \ldots S_k$ of $\mathcal{S}$.[2] It is *regular* if the language consisting of words derivable from any $R \in \mathbf{R}$ is regular. Assume that R is derivable from itself.

[2] If $k = 0$, then the RHS (right hand side) of each of the rules represents the empty word ε.

A *role inclusion axiom* (RIA) has the form $S_1 \circ \cdots \circ S_k \sqsubseteq R$, where $k \geq 0$ and $S_1, \ldots, S_k, R$ are basic roles. If $k = 0$, then the LHS (left hand side) of the inclusion stands for ε.

A *(regular) RBox* is a finite set $\mathcal{R}$ of RIAs such that $\mathcal{S} = \{R \to S_1 \ldots S_k \mid (S_1 \circ \cdots \circ S_k \sqsubseteq R) \in \mathcal{R}\}$ is a regular and symmetric semi-Thue system with the property that only ε and words with length 1 can be derived from any simple role $R \in \mathbf{R}$. An RBox is allowed for a DL $\mathcal{L}$ if it uses inverse roles only when they are allowed for $\mathcal{L}$. Since it is undecidable whether a context-free semi-Thue system is regular, we assume that each RBox $\mathcal{R}$ is accompanied by a mapping $\pi_{\mathcal{R}}$ that associates each $R \in \mathbf{R}$ with a regular expression $\pi_{\mathcal{R}}(R)$ that generates the set of words derivable from R using the rules of the corresponding semi-Thue system.

If $S_1 \circ \cdots \circ S_k \sqsubseteq R$ is a RIA of $\mathcal{R}$, then we call R an *intensional predicate* specified by $\mathcal{R}$. An interpretation $\mathcal{I}$ *validates* a RIA $S_1 \circ \cdots \circ S_k \sqsubseteq R$ if $(S_1 \circ \cdots \circ S_k)^{\mathcal{I}} \subseteq R^{\mathcal{I}}$. It is a *model* of an RBox $\mathcal{R}$ if it validates all RIAs of $\mathcal{R}$.

2.2 TBoxes

A *TBox axiom* (or *terminological axiom*) is either a general concept inclusion (GCI) $C \sqsubseteq D$ or a concept equivalence $C \doteq D$. A concept equivalence $A \doteq D$ (where $A \in \mathbf{C}$) is called a *concept definition*. A *TBox* is a finite set of TBox axioms. It is allowed for a DL $\mathcal{L}$ if it uses only concepts of $\mathcal{L}$. An interpretation $\mathcal{I}$ *validates* $C \sqsubseteq D$ (resp. $C \doteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (resp. $C^{\mathcal{I}} = D^{\mathcal{I}}$). It is a *model* of a TBox $\mathcal{T}$ if it validates all axioms of $\mathcal{T}$.

A TBox $\mathcal{T}$ is *acyclic* if there exist concept names $A_1, \ldots, A_n$ such that $\mathcal{T}$ consists of n axioms and the i-th axiom of $\mathcal{T}$ is of the form $A_i \doteq C$, $C \sqsubseteq A_i$ or $A_i \sqsubseteq C$, where C does not use the concept names $A_i, \ldots, A_n$. The concept names $A_1, \ldots, A_n$ are called *intensional predicates* specified by $\mathcal{T}$.

A TBox $\mathcal{T}$ is called a *simple stratified TBox* if there exists a partition $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$ of $\mathcal{T}$, called a *stratification* of $\mathcal{T}$, such that, for each $1 \leq i \leq n$, $\mathcal{T}_i = \{C_{i,j} \sqsubseteq A_{i,j} \mid 1 \leq j \leq n_i\}$, where each $A_{i,j}$ is a concept name that does not occur in $\mathcal{T}_1, \ldots, \mathcal{T}_{i-1}$ and may occur at the LHS of $\sqsubseteq$ in the axioms of $\mathcal{T}_i$ only under the scope of $\sqcap$, $\sqcup$ and $\exists$. The concept names $A_{i,j}$, for $1 \leq i \leq n$ and $1 \leq j \leq n_i$, are called *intensional predicates* specified by $\mathcal{T}$.

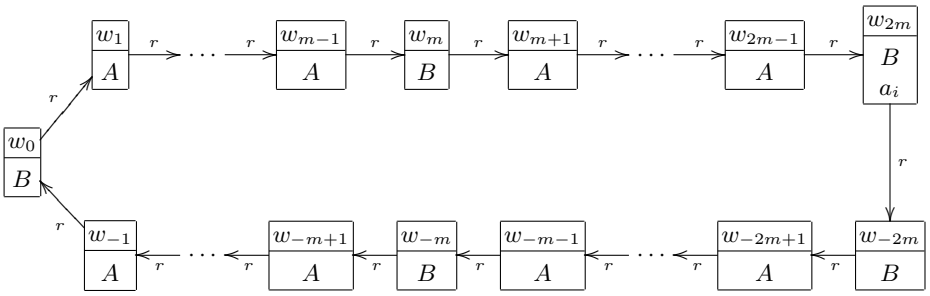
Note that negation ($\neg$) is allowed at the LHS of $\sqsubseteq$ in GCIs of a simple stratified TBox, but it can be applied only to concepts that do not use the predicates defined in the current or later strata.

3 The First Result

In this section, we prove the following theorem:

Theorem 1. *There is no concept of $\mathcal{ALCIOQUS}elf_{trans}$ equivalent to the concept $C = \exists((r \circ A?)^* \circ r \circ B? \circ r \circ A?).\top$ or $C = \exists((r \circ A?)^* \circ r \circ (\neg A)? \circ r \circ A?).\top$ of $\mathcal{ALC}_{reg}$.*

To prove this theorem we will use a family of interpretations $\mathcal{I}_m = \langle \Delta^{\mathcal{I}_m}, \cdot^{\mathcal{I}_m} \rangle$, $m > 1$, illustrated and specified as follows:



- $\Delta^{\mathcal{I}_m} = \{w_{-2m}, w_{-2m+1}, \ldots, w_{2m}\}$,
- $r^{\mathcal{I}_m} = \{\langle w_i, w_{i+1}\rangle, \langle w_{2m}, w_{-2m}\rangle \mid -2m \leq i < 2m\}$,
- $s^{\mathcal{I}_m} = \emptyset$ for $s \in \mathbf{R}_+ - \{r\}$,
- $B^{\mathcal{I}_m} = \{w_{-2m}, w_{-m}, w_0, w_m, w_{2m}\}$,
- $A^{\mathcal{I}_m} = \Delta^{\mathcal{I}_m} - B^{\mathcal{I}_m}$,
- $C^{\mathcal{I}_m} = \emptyset$ for $C \in \mathbf{C} - \{A, B\}$,
- $a^{\mathcal{I}_m} = w_{2m}$ for $a \in \mathbf{I}$.

Note that $|\Delta^{\mathcal{I}_m}| = 4m + 1$. Comparing $\mathcal{I}_m$ with the structure $\mathcal{A}_m$ used in [2], note that the domain of $\mathcal{A}_m$ has the size $2m+1$, $\mathcal{A}_m$ does not deal with nominals, and only one propositional variable is interpreted in $\mathcal{A}_m$ as a non-empty subset of the domain.

Observe that, for C being one of the two concepts mentioned in Theorem 1, $w_0 \in C^{\mathcal{I}_m}$ but $w_m \notin C^{\mathcal{I}_m}$. The structure of the proof of Theorem 1 is as follows. Given any concept D of $\mathcal{ALCIOQUSelf}_{trans}$, we first transform it to a concept D_2 of $\mathcal{ALCIO}_{trans}$ over the signature $\{r, A, B, a\}$ such that $D_2^{\mathcal{I}_m} = D^{\mathcal{I}_m}$ for all $m > 1$ (see Lemma 1). We then transform D_2 to a concept D_3 such that $D_3^{\mathcal{I}_m} = D_2^{\mathcal{I}_m}$ for all $m > 1$, the $*$ operator is used only for r^n and $\overline{r}^n$ for some n (see Lemma 2), and for every subconcept $\exists R.D_3'$ or $\forall R.D_3'$ of D_3, R is of the form r, $\overline{r}$, $(r^n)^*$ or $(\overline{r}^n)^*$ for some $n \geq 1$ (see Lemma 3). Next, we show that there exists $m > 1$ such that $w_0 \in D_3^{\mathcal{I}_m} \Leftrightarrow w_m \in D_3^{\mathcal{I}_m}$ (see Lemma 4). Thus, for that m, $C^{\mathcal{I}_m} \neq D_3^{\mathcal{I}_m}$, and therefore, C is not equivalent to D (since $D_3^{\mathcal{I}_m} = D_2^{\mathcal{I}_m} = D^{\mathcal{I}_m}$).

Lemma 1. *For any concept C of $\mathcal{ALCIOQUSelf}_{trans}$, there exists a concept D of $\mathcal{ALCIO}_{trans}$ over the signature $\{r, A, B, a\}$ such that $D^{\mathcal{I}_m} = C^{\mathcal{I}_m}$ for all $m > 1$.*

Proof. Let D be the concept obtained from C by:

- replacing every subconcept

- $\geq n\, R.E$, where $n \geq 2$, by $\bot$,
- $\geq 1\, R.E$ by $\exists R.E$,
- $\geq 0\, R.E$ by $\top$,
- $\leq n\, R.E$, where $n \geq 1$, by $\top$,
- $\leq 0\, R.E$ by $\forall R.\neg E$,
- $\exists U.E$ by $\exists r^*.E$,
- $\forall U.E$ by $\forall r^*.E$,
- $\exists R.\mathcal{S}elf$ by $\bot$,

– replacing every concept name different from A and B by $\bot$,
– replacing every nominal $\{b\}$, where $b \neq a$, by $\{a\}$,
– replacing every role name s different from r by $\emptyset$,
– repeatedly replacing every role $\emptyset \sqcup R$ or $R \sqcup \emptyset$ by R, every role $\emptyset^*$ by ε, and every role $\overline{\emptyset}$, $\emptyset \circ R$ or $R \circ \emptyset$ by $\emptyset$,
– replacing every subconcept $\exists\emptyset.E$ by $\bot$, and every $\forall\emptyset.E$ by $\top$.

It is easy to see that D satisfies the properties mentioned in the lemma. □

We treat a word $R_1 \dots R_k$ over the alphabet $\{r, \overline{r}\}$ as the role $R_1 \circ \dots \circ R_k$, and by R^n we denote the composition of n copies of R. Thus, $R^0 = \varepsilon$. Conversely, a role R without tests that uses only basic roles r and $\overline{r}$ is treated as a regular expression over the alphabet $\{r, \overline{r}\}$ (where $\sqcup$ stands for $\cup$, and $\circ$ for ;). For such a role R, by $\mathcal{L}(R)$ we denote the regular language generated by R. For a word R over the alphabet $\{r, \overline{r}\}$, by $|R|$ we denote the *length* of R (defined in the usual way), and by $||R||$ we denote the *norm* of R, which is defined as follows: $||\varepsilon|| = 0$, $||r|| = 1$, $||\overline{r}|| = -1$, $||RS|| = ||R|| + ||S||$. Observe that, for words R and S over the alphabet $\{r, \overline{r}\}$, if $||R|| = ||S||$, then $R^{\mathcal{I}_m} = S^{\mathcal{I}_m}$ for all $m > 1$.

Lemma 2. *Let R be a role without tests that uses only basic roles r and $\overline{r}$. Then, there exists a role S such that $S^{\mathcal{I}_m} = R^{\mathcal{I}_m}$ for all $m > 1$ and the $*$ operator can be used in S only for r^n and $\overline{r}^n$ for some n.*

Proof. Since $\mathcal{L}(R)$ is a regular language, by the pumping lemma, there exists an integer $p > 0$ such that every word from $\mathcal{L}(R)$ of length at least p can be represented as xyz such that $|y| > 0$, $|xy| \leq p$ and $xy^iz \in \mathcal{L}(R)$ for all $i \geq 0$.

Let $n = p(p-1)\cdots 2 \cdot 1$ and let $\mathcal{L}'$ be the language obtained from $\mathcal{L}(R)$ by deleting all words y such that there exists $x \in \mathcal{L}(R)$ with $|x| < |y|$ and $||x|| = ||y||$. By $\mathit{pumping}(x, y, z)$ we denote the formula

$$xyz \in \mathcal{L}' \wedge |y| > 0 \wedge |xy| \leq p \wedge \forall i \geq 0\ xy^iz \in \mathcal{L}(R).$$

Observe that, if $w' = xyz \in \mathcal{L}'$ and $\mathit{pumping}(x, y, z)$ holds, then:

– $||y|| \neq 0$ because otherwise we would have $xz \in \mathcal{L}(R)$, $|xz| < |w'|$ and $||xz|| = ||w'||$, which contradict the definition of $\mathcal{L}'$;
– if $||y|| > 0$ then, for all $i \geq 0$, there exists $u \in \mathcal{L}(R)$ with $||u|| = ||w'(r^n)^i||$;
– if $||y|| < 0$ then, for all $i \geq 0$, there exists $u \in \mathcal{L}(R)$ with $||u|| = ||w'(\overline{r}^n)^i||$.

Denote this observation by $(\star)$. For each integer j, $0 \leq j < n$, let

$$K_j^+ = \{||xyz|| : \mathit{pumping}(x, y, z),\ n \,|\, (||xyz|| - j) \text{ and } ||y|| > 0\}$$
$$K_j^- = \{||xyz|| : \mathit{pumping}(x, y, z),\ n \,|\, (||xyz|| - j) \text{ and } ||y|| < 0\}.$$

For intuition, informally, we intend to define S to be the role

$$\bigsqcup \mathcal{S}_1 \sqcup ((\bigsqcup \mathcal{S}_2) \circ (r^n)^*) \sqcup ((\bigsqcup \mathcal{S}_3) \circ (\overline{r}^n)^*), \tag{1}$$

where $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ are the finite sets of words over the alphabet $\{r, \overline{r}\}$ constructed as follows:

- $\mathcal{S}_1 := \{x \in \mathcal{L}' : |x| < p\}$, $\mathcal{S}_2 := \emptyset$, $\mathcal{S}_3 := \emptyset$;
- for each j from 0 to $n-1$ do
  - if $K_j^+ \neq \emptyset$ then
    - if K_j^+ does not have a minimum then add r^j to both $\mathcal{S}_2$ and $\mathcal{S}_3$;
    - else: let $k = \min K_j^+$, if $k \geq 0$ then $\mathcal{S}_2 := \mathcal{S}_2 \cup \{r^k\}$ else $\mathcal{S}_2 := \mathcal{S}_2 \cup \{(\overline{r})^{-k}\}$ (for this second case, notice that $-k$ is a positive integer);
  - if $K_j^- \neq \emptyset$ then
    - if K_j^- does not have a maximum then add r^j to both $\mathcal{S}_2$ and $\mathcal{S}_3$;
    - else: let $k = \max K_j^-$, if $k \geq 0$ then $\mathcal{S}_3 := \mathcal{S}_3 \cup \{r^k\}$ else $\mathcal{S}_3 := \mathcal{S}_3 \cup \{(\overline{r})^{-k}\}$.

Formally, we define S to be the role obtained from (1) by deleting any i-th main disjunct such that $\mathcal{S}_i$ is empty, for $i \in \{1, 2, 3\}$. To prove that $S^{\mathcal{I}_m} = R^{\mathcal{I}_m}$ for all $m > 1$ it is sufficient to show that:

1. if $w \in \mathcal{L}(S)$, then there exists $u \in \mathcal{L}(R)$ such that $||u|| = ||w||$,
2. if $w \in \mathcal{L}(R)$, then there exists $u \in \mathcal{L}(S)$ such that $||u|| = ||w||$.

Consider the assertion (1) and let $w \in \mathcal{L}(S)$. There are the following cases:

- Case $w \in \mathcal{S}_1$: We have that $w \in \mathcal{L}' \subseteq \mathcal{L}(R)$. Just take $u = w$.
- Case $w = r^j(r^n)^h$, $K_j^+ \neq \emptyset$ and K_j^+ does not have a minimum: Thus, there exists $w' = xyz \in \mathcal{L}'$ such that $pumping(x, y, z)$ holds, $||y|| > 0$ and $||w'|| = j + n \cdot h'$ for some $h' < h$. By $(\star)$, there exists $u \in \mathcal{L}(R)$ such that $||u|| = ||w||$.
- Case $w = r^k(r^n)^h$, $K_j^+ \neq \emptyset$, $k = \min K_j^+$ and $k \geq 0$: Thus, there exists $w' = xyz \in \mathcal{L}'$ such that $pumping(x, y, z)$ holds, $||y|| > 0$ and $||w'|| = k$. By $(\star)$, there exists $u \in \mathcal{L}(R)$ such that $||u|| = ||w||$.
- Case $w = (\overline{r})^{-k}(r^n)^h$, $K_j^+ \neq \emptyset$, $k = \min K_j^+$ and $k < 0$: Thus, there exists $xyz \in \mathcal{L}'$ such that $pumping(x, y, z)$ holds, $||y|| > 0$ and $||w'|| = k$. Notice that $||(\overline{r})^{-k}|| = k$. By $(\star)$, there exists $u \in \mathcal{L}(R)$ such that $||u|| = ||w||$.
- Case $w = r^j(r^n)^h$, $K_j^- \neq \emptyset$ and K_j^- does not have a maximum: Thus, there exists $w' = xyz \in \mathcal{L}'$ such that $pumping(x, y, z)$ holds, $||y|| < 0$ and $||w'|| = j + n \cdot h'$ for some $h' > h$. By $(\star)$, there exists $u \in \mathcal{L}(R)$ such that $||u|| = ||w||$.
- The four previous cases are related to $\mathcal{S}_2$. The four remaining cases, which are related to $\mathcal{S}_3$, can be dealt with in a similar way.

Consider the assertion (2) and let $w \in \mathcal{L}(R)$. There exists $w' \in \mathcal{L}'$ such that $||w'|| = ||w||$. If $|w'| < p$, then $w' \in \mathcal{S}_1$ and we can just take $u = w'$. Suppose $|w'| \geq p$. Thus, w' can be represented as xyz such that $pumping(x, y, z)$ holds. There are the following cases:

- Case $||y|| > 0$: There exists $0 \leq j < n$ such that $||w'|| \in K_j^+$ and $||w'|| = j + n \cdot i$ for some integer i. Consider the following subcases.
  - Case K_j^+ does not have a minimum: Thus, $r^j \in \mathcal{S}_2$. Taking $u = r^j(r^n)^i$, we have that $u \in \mathcal{L}(S)$ and $||u|| = ||w'|| = ||w||$.
  - Case $k = \min K_j^+$ and $k \geq 0$: Thus, $r^k \in \mathcal{S}_2$. Observe that $||w'|| \geq k$ and $n \,|\, (||w'|| - k)$. Taking $u = r^{||w'||}$, we have that $u \in \mathcal{L}(S)$ and $||u|| = ||w'|| = ||w||$.
  - Case $k = \min K_j^+$ and $k < 0$: Thus, $(\overline{r})^{-k} \in \mathcal{S}_2$. Observe that $||w'|| \geq k$ and $n \,|\, (||w'|| - k)$. Taking $u = (\overline{r})^{-k}(r^{||w'||-k})$, we have that $u \in \mathcal{L}(S)$ and $||u|| = ||w'|| = ||w||$.
- The case when $||y|| < 0$ is dual to the above case and can be dealt with analogously. □

Let $\mathfrak{C}$ denote the set of concepts C of $\mathcal{ALCIO}_{trans}$ over the signature $\{r, A, B, a\}$ such that, for every subconcept $\exists R.D$ or $\forall R.D$ of C, R is of the form r, $\overline{r}$, $(r^n)^*$ or $(\overline{r}^n)^*$ for some $n \geq 1$.

Lemma 3. *For any concept C of $\mathcal{ALCIO}_{trans}$ over the signature $\{r, A, B, a\}$, there exists a concept $D \in \mathfrak{C}$ such that $D^{\mathcal{I}_m} = C^{\mathcal{I}_m}$ for all $m > 1$.*

Proof. Let E be the concept obtained from C by replacing every role R by a role S that satisfies the conditions mentioned in Lemma 2. We have $E^{\mathcal{I}_m} = C^{\mathcal{I}_m}$ for all $m > 1$. Then, let D be obtained from E by repeatedly applying the following transformations:

$$\begin{array}{cc} \exists(R \sqcup S).F \equiv \exists R.F \sqcup \exists R.F & \forall(R \sqcup S).F \equiv \forall R.F \sqcap \forall R.F \\ \exists(R \circ S).F \equiv \exists R.\exists S.F & \forall(R \circ S).F \equiv \forall R.\forall S.F \\ \exists\varepsilon.F \equiv F & \forall\varepsilon.F \equiv F. \end{array}$$

It is clear that $D \in \mathfrak{C}$ and $D^{\mathcal{I}_m} = E^{\mathcal{I}_m} = C^{\mathcal{I}_m}$ for all $m > 1$. □

For a concept $C \in \mathfrak{C}$, by $n_r(C)$ we denote the number of occurrences of $\exists r$, $\exists \overline{r}$, $\forall r$ and $\forall \overline{r}$ in C.

Lemma 4. *For any concept $C \in \mathfrak{C}$ and integers m and k such that $m > 1$, $4m + 1$ is prime and $n_r(C) < m - |k|$, we have $w_k \in C^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in C^{\mathcal{I}_m}$.*

Proof. This proof is similar to the one of [2, Lemma 3]. The intuition is as follows:

- a concept C' can distinguish w_k and w_{k+m} only if $n_r(C')$ is large enough so that the checking can recognize that the neighborhood of w_k differs from the corresponding neighborhood of w_{k+m}, in particular, to recognize that the first one contains w_{m+1} (resp. w_{-m-1}) and the second one contains w_{-2m} (resp. w_{2m}); the reason is that, since $4m + 1$ is prime, either $((r^n)^*)^{\mathcal{I}_m} = \Delta^{\mathcal{I}_m} \times \Delta^{\mathcal{I}_m}$ or $\langle w_i, w_j \rangle \in ((r^n)^*)^{\mathcal{I}_m}$ iff $j = i$;
- since $n_r(C) < m - |k|$, C cannot distinguish w_k and w_{k+m}.

Observe that $-m < k < m$. We prove this lemma by induction on the structure of C. The cases when C is A, B, $\top$ or $\bot$ are trivial. The cases when C is of the form $D \sqcap E$ or $\forall R.D$ are reduced to the cases of $\neg(\neg D \sqcup \neg E)$ and $\neg\exists R.\neg D$, respectively.

- Case $C = \{a\}$: Since $a^{\mathcal{I}_m} = w_{2m}$, $w_k \notin C^{\mathcal{I}_m}$ and $w_{k+m} \notin C^{\mathcal{I}_m}$.
- Case $C = \neg D$: We have $n_r(D) = n_r(C)$. By induction, $w_k \in D^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in D^{\mathcal{I}_m}$, and hence, $w_k \in C^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in C^{\mathcal{I}_m}$.
- Case $C = D \sqcup E$: We have $n_r(D) \leq n_r(C)$ and $n_r(E) \leq n_r(C)$. By induction, $w_k \in D^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in D^{\mathcal{I}_m}$ and $w_k \in E^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in E^{\mathcal{I}_m}$, which imply that $w_k \in C^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in C^{\mathcal{I}_m}$.
- Case $C = \exists r.D$: We have $n_r(D) = n_r(C) - 1 < m - |k| - 1 \leq m - |k+1|$. By induction, $w_{k+1} \in D^{\mathcal{I}_m} \Leftrightarrow w_{k+1+m} \in D^{\mathcal{I}_m}$. Hence, $w_k \in (\exists r.D)^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in (\exists r.D)^{\mathcal{I}_m}$, which means $w_k \in C^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in C^{\mathcal{I}_m}$.
- Case $C = \exists \overline{r}.D$: We have $n_r(D) = n_r(C) - 1 < m - |k| - 1 \leq m - |k-1|$. By induction, $w_{k-1} \in D^{\mathcal{I}_m} \Leftrightarrow w_{k-1+m} \in D^{\mathcal{I}_m}$. Similarly to the previous case, this implies that $w_k \in (\exists \overline{r}.D)^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in (\exists \overline{r}.D)^{\mathcal{I}_m}$, which means $w_k \in C^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in C^{\mathcal{I}_m}$.
- Case $C = \exists (r^n)^*.D$ and $(4m+1) | n$: We have $\langle w_i, w_j \rangle \in ((r^n)^*)^{\mathcal{I}_m}$ iff $j = i$. Hence,
$$w_k \in (\exists (r^n)^*.D)^{\mathcal{I}_m} \Leftrightarrow w_k \in D^{\mathcal{I}_m}$$
$$w_{k+m} \in (\exists (r^n)^*.D)^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in D^{\mathcal{I}_m}.$$
We have $n_r(D) = n_r(C)$. By induction, $w_k \in D^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in D^{\mathcal{I}_m}$. Therefore, $w_k \in C^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in C^{\mathcal{I}_m}$.
- Case $C = \exists (r^n)^*.D$ and $(4m+1) \not| \ n$: Since $4m+1$ is prime, $0, n, 2n, 3n, \ldots, (4m)n$ have all $4m+1$ different residues modulo $4m+1$. Hence, $\langle w_i, w_j \rangle \in ((r^n)^*)^{\mathcal{I}_m}$ for all $w_i, w_j \in \Delta^{\mathcal{I}_m}$, and
$$w_k \in (\exists (r^n)^*.D)^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in (\exists (r^n)^*.D)^{\mathcal{I}_m},$$
because they are both equivalent to that there exists $w_j \in D^{\mathcal{I}_m}$. Therefore,
$$w_k \in C^{\mathcal{I}_m} \Leftrightarrow w_{k+m} \in C^{\mathcal{I}_m}.$$
- The case $C = \exists (\overline{r}^n)^*.D$ is similar to the two previous cases. □

We now recall and prove Theorem 1.

Theorem 1 *There is no concept of $\mathcal{ALCIOQUSelf}_{trans}$ equivalent to the concept $C = \exists((r \circ A?)^* \circ r \circ B? \circ r \circ A?).\top$ or $C = \exists((r \circ A?)^* \circ r \circ (\neg A)? \circ r \circ A?).\top$ of $\mathcal{ALC}_{reg}$.*

Proof. For a contradiction, suppose D is a concept of $\mathcal{ALCIOQUSelf}_{trans}$ equivalent to C. By Lemma 1, there exists a concept D_2 of $\mathcal{ALCIO}_{trans}$ over the signature $\{r, A, B, a\}$ such that $D_2^{\mathcal{I}_m} = D^{\mathcal{I}_m}$ for all $m > 1$. By Lemma 3, there exists a concept $D_3 \in \mathfrak{C}$ such that $D_3^{\mathcal{I}_m} = D_2^{\mathcal{I}_m}$ for all $m > 1$. Let m be an integer such that $m > n_r(D_3)$ and $4m+1$ is prime. By Lemma 4, $w_0 \in D_3^{\mathcal{I}_m} \Leftrightarrow w_m \in D_3^{\mathcal{I}_m}$. This contradicts the facts that $D_3^{\mathcal{I}_m} = D_2^{\mathcal{I}_m} = D^{\mathcal{I}_m} = C^{\mathcal{I}_m}$, $w_0 \in C^{\mathcal{I}_m}$ and $w_m \notin C^{\mathcal{I}_m}$. □

Corollary 1. *For any* $\Phi \subseteq \{\mathcal{I}, \mathcal{O}, \mathcal{Q}, \mathcal{U}, \mathit{Self}\}$, $(\mathcal{ALC}{+}\Phi)_{trans}$ *is weaker than* $(\mathcal{ALC}{+}\Phi)_{reg}$ *in expressing concepts.*

4 Dealing with RBoxes and TBoxes

The result of the previous section roughly states that, without using RBoxes and TBoxes, it is hard to eliminate tests, at least it is impossible to eliminate tests from $\mathcal{ALCIOQUSelf}_{reg}$ without decreasing the expressive power. As expected, using acyclic TBoxes that consist only of concept definitions do not help in expressing tests. The first result of this section states that using RBoxes and acyclic TBoxes that are defined more liberally as in Section 2 does not help either. The second result states that, however, using simple stratified TBoxes under the stratified semantics on the background, it is possible to express every concept by another without tests. The third result states that tests can be eliminated from the deterministic Horn fragment of $\mathcal{ALC}_{reg}$. Due to the lack of space, proofs of these results are provided only in the long version [11] of the current paper.

4.1 The Case with RBoxes and Acyclic TBoxes

We say that a concept C is *inexpressible in a DL* $\mathcal{L}$ *even when using RBoxes and acyclic TBoxes* if, for every concept D, every RBox $\mathcal{R}$ and every acyclic TBox $\mathcal{T}$ of $\mathcal{L}$ such that the intensional predicates specified by $\mathcal{R}$ and $\mathcal{T}$ do not occur in C, there exists a model $\mathcal{I}$ of $\mathcal{R}$ and $\mathcal{T}$ such that $C^{\mathcal{I}} \neq D^{\mathcal{I}}$.

Proposition 1. *The concept* $C = \exists((r \circ A?)^* \circ r \circ B? \circ r \circ A?).\top$ *or* $C = \exists((r \circ A?)^* \circ r \circ (\neg A)? \circ r \circ A?).\top$ *of* $\mathcal{ALC}_{reg}$ *is inexpressible in* $\mathcal{ALCIOQUSelf}_{trans}$ *even when using RBoxes and acyclic TBoxes.*

4.2 Eliminating Tests from Concepts by Simple Stratified TBoxes

Let $\mathcal{T}$ be a simple stratified TBox. An interpretation $\mathcal{I}$ is called a *standard model* of $\mathcal{T}$ (under the stratified semantics) if there exist a partition $(\mathcal{T}_1, \ldots, \mathcal{T}_n)$ of $\mathcal{T}$ and interpretations $\mathcal{J}_0, \ldots, \mathcal{J}_n$ such that:

- $\mathcal{T}_i = \{C_{i,j} \sqsubseteq A_{i,j} \mid 1 \leq j \leq n_i\}$ for $1 \leq i \leq n$,
- $\mathcal{J}_n = \mathcal{I}$ and $\Delta^{\mathcal{J}_i} = \Delta^{\mathcal{I}}$ for all $0 \leq i < n$,
- $x^{\mathcal{J}_0} = x^{\mathcal{I}}$ for all $x \in \Sigma - \{A_{i,j} \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq n_i\}$,
- for each $1 \leq i \leq n$, $x^{\mathcal{J}_i} = x^{\mathcal{J}_{i-1}}$ for all $x \in \Sigma - \{A_{i',j} \mid i \leq i' \leq n, 1 \leq j \leq n_{i'}\}$ and $A_{i,j}^{\mathcal{J}_i}$, for $1 \leq j \leq n_i$, are the smallest subsets of $\Delta^{\mathcal{J}_i}$ such that $A_{i,j}^{\mathcal{J}_i} = C_{i,j}^{\mathcal{J}_i}$.

It can be shown that, for every interpretation $\mathcal{J}_0$, there exists a unique standard model $\mathcal{I}$ of $\mathcal{T}$ such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}_0}$ and $x^{\mathcal{I}} = x^{\mathcal{J}_0}$ for all $x \in \Sigma - \{A_{i,j} \mid 1 \leq i \leq n$ and $1 \leq j \leq n_i\}$. We call it the *standard model of* $\mathcal{T}$ *based on* $\mathcal{J}_0$.

In what follows, let $\Phi \subseteq \{\mathcal{I}, \mathcal{O}, \mathcal{Q}, \mathcal{U}, \mathit{Self}\}$ (in general, extending Φ with other features does not affect Proposition 2 given below). Let C be a concept of $(\mathcal{ALC}{+}\Phi)_{reg}$, D a concept and $\mathcal{T}$ a simple stratified TBox of $(\mathcal{ALC}{+}\Phi)_{trans}$ such

that the intensional predicates specified by $\mathcal{T}$ do not occur in C. We say that C is *expressed by D and $\mathcal{T}$ under the stratified semantics* if, for every standard model $\mathcal{I}$ of $\mathcal{T}$, $C^{\mathcal{I}} = D^{\mathcal{I}}$.

Proposition 2. *Every concept of $(\mathcal{ALC}{+}\Phi)_{reg}$ can be expressed by a concept and a simple stratified TBox of $(\mathcal{ALC}{+}\Phi)_{trans}$ under the stratified semantics.*

4.3 Eliminating Tests from Horn TBoxes

The previous subsection deals with eliminating tests from a standing alone concept by using a simple stratified TBox under the stratified semantics. Roughly speaking, it suggests that tests in PDL-like roles can be eliminated by using fixpoints outside roles. The result of this subsection states that tests can be eliminated from TBoxes of the deterministic Horn fragment of $\mathcal{ALC}_{reg}$. This is possible because the traditional semantics of such TBoxes has a fixpoint characterization.

A role can be treated as a regular expression over the alphabet $\mathbf{R}_+ \cup \{C? \mid C \text{ is a concept}\}$, where $\sqcup$ and $\circ$ stand for $\cup$ and semicolon, respectively. Conversely, a word over this alphabet can be treated as a role. Given a role R, let $\mathcal{L}(R)$ denote the regular language generated by R and let $\forall\exists R.C$ be a new concept constructor whose semantics in an interpretation $\mathcal{I}$ is specified as follows:

$$(\forall\exists R.C)^{\mathcal{I}} = \bigcap\{(\forall S.\exists S'.C)^{\mathcal{I}} \mid SS' \in \mathcal{L}(R)\}.$$

Observe that, if $R \in \mathbf{R}_+$, then $\forall\exists R.C \equiv \forall R.C \sqcap \exists R.C$.

The deterministic Horn fragment of $\mathcal{ALC}_{reg}$, denoted by D-Horn-$\mathcal{ALC}_{reg}$, is designed with the intention to be (probably) the most expressive fragment of $\mathcal{ALC}_{reg}$ that has a PTIME data complexity (under the traditional semantics).

A *D-Horn-$\mathcal{ALC}_{reg}$ TBox axiom* is an expression of the form $C_l \sqsubseteq C_r$, where C_l and C_r are concepts defined by the following BNF grammar, with $A \in \mathbf{C}$ and $s \in \mathbf{R}_+$:

$$C_l ::= \top \mid A \mid C_l \sqcap C_l \mid C_l \sqcup C_l \mid \exists R_l.C_l \mid \forall\exists R_l.C_l \tag{2}$$
$$R_l ::= s \mid R_l \circ R_l \mid R_l \sqcup R_l \mid R_l^* \mid C_l? \tag{3}$$
$$C_r ::= \top \mid \bot \mid A \mid \neg C_l \mid C_r \sqcap C_r \mid \neg C_l \sqcup C_r \mid \exists R_r.C_r \mid \forall R_l.C_r \tag{4}$$
$$R_r ::= s \mid R_r \circ R_r \mid C_r? \tag{5}$$

A *D-Horn-$\mathcal{ALC}_{reg}$ TBox* is a finite set of D-Horn-$\mathcal{ALC}_{reg}$ TBox axioms.

Remark 1. A (reduced) *ABox* is a finite set of *assertions* of the form $A(a)$, $\neg A(a)$ or $r(a, b)$ (where $A \in \mathbf{C}$ and $r \in \mathbf{R}_+$). A *knowledge base* in D-Horn-$\mathcal{ALC}_{reg}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ consisting of a D-Horn-$\mathcal{ALC}_{reg}$ TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The notion of whether an interpretation is a *model* of an ABox or a knowledge base is defined in the usual way. A knowledge base is *satisfiable* if it has a model. It can be proved that checking whether a given knowledge base $\langle \mathcal{T}, \mathcal{A} \rangle$ in D-Horn-$\mathcal{ALC}_{reg}$

is satisfiable is solvable in polynomial time in the size of the ABox $\mathcal{A}$.[3] That is, D-Horn-$\mathcal{ALC}_{reg}$ has a PTIME data complexity. If $\forall\exists$ in (2) is replaced by $\forall$, then instead of D-Horn-$\mathcal{ALC}_{reg}$ we obtain the general Horn fragment of $\mathcal{ALC}_{reg}$ with a NP-hard data complexity for the satisfiability problem.[4] □

The following theorem states that tests can be eliminated from D-Horn-$\mathcal{ALC}_{reg}$.

Theorem 2. *For every D-Horn-$\mathcal{ALC}_{reg}$ TBox $\mathcal{T}$ over a signature Σ, there exists a D-Horn-$\mathcal{ALC}_{reg}$ TBox $\mathcal{T}'$ without tests over a signature $\Sigma' \supseteq \Sigma$ such that:*

1. *for every model $\mathcal{I}$ of $\mathcal{T}$, there exists a model $\mathcal{I}'$ of $\mathcal{T}'$ such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$ and $x^{\mathcal{I}} = x^{\mathcal{I}'}$ for all $x \in \Sigma$,*
2. *for every model $\mathcal{I}'$ of $\mathcal{T}'$, the interpretation $\mathcal{I}$ over Σ specified by $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$ and $x^{\mathcal{I}} = x^{\mathcal{I}'}$ for all $x \in \Sigma$ is a model of $\mathcal{T}$.*

5 Conclusions

Generalizing the result and method of Berman and Paterson [2], we have proved that there is a concept of $\mathcal{ALC}_{reg}$ that is not equivalent to any concept of the DL that extends $\mathcal{ALC}_{trans}$ with inverse roles, nominals, qualified number restrictions, the universal role and local reflexivity of roles. This implies, among others, that CPDL (Converse-PDL) is more expressive than Test-Free CPDL, and GCPDL (Graded Converse-PDL) is more expressive than Test-Free GCPDL. Extending our result by applying the technique of [2], it can also be proved that CPDL$_{n+1}$ (CPDL with at most $n+1$ levels of nesting of tests) is more expressive than CPDL$_n$, and similarly for GCPDL.

The other results of this paper state that, on the other hand, using simple stratified TBoxes under the stratified semantics on the background, it is possible to express every concept by another without tests. Furthermore, tests can be eliminated from the deterministic Horn fragment D-Horn-$\mathcal{ALC}_{reg}$ of $\mathcal{ALC}_{reg}$. If one extends D-Horn-$\mathcal{ALC}_{reg}$ with other features (e.g., $\mathcal{I}$, $\mathcal{O}$, $\mathcal{Q}$, $\mathcal{U}$ and $\mathcal{S}elf$) appropriately so that the resulting language still has a PTIME data complexity (cf. Horn-$\mathcal{SHIQ}$ [9], Horn-$\mathcal{SROIQ}$ [15] and Horn-DL [14]), then our elimination technique (presented in the long version [11] of the current paper) can still be applied. Besides, it is hard to define a fragment of $\mathcal{ALC}_{reg}$ that is more expressive than D-Horn-$\mathcal{ALC}_{reg}$ and still has a PTIME data complexity under the traditional semantics. So, we have a tendency to claim that tests can be eliminated from tractable Horn fragments of PDL-like logics.

[3] A more general result was proved in [13] for D-Horn-CPDL$_{reg}$, which extends D-Horn-$\mathcal{ALC}_{reg}$ with inverse roles and regular RBoxes.

[4] The hardness was shown for the general Horn fragment of $\mathcal{ALC}$ [12].

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
2. Berman, F., Paterson, M.: Propositional dynamic logic is weaker without tests. Theor. Comput. Sci. **16**, 321–328 (1981)
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press (2002)
4. Divroodi, A., Nguyen, L.: On bisimulations for description logics. Inf. Sci. **295**, 465–493 (2015)
5. Fischer, M., Ladner, R.: Propositional dynamic logic of regular programs. J. Comput. Syst. Sci. **18**(2), 194–211 (1979)
6. Ginsburg, S.: The Mathematical Theory of Context-Free Languages. McGraw-Hill, Inc. (1966)
7. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)
8. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proceedings of KR. pp. 57–67. AAAI Press (2006)
9. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive Datalog. J. Autom. Reasoning **39**(3), 351–384 (2007)
10. Lutz, C., Piro, R., Wolter, F.: Description logic TBoxes: Model-theoretic characterizations and rewritability. In: Proceedings of IJCAI. pp. 983–988 (2011)
11. Nguyen, L.: A long version of the current paper. Available at `http://www.mimuw.edu.pl/~nguyen/EPT.pdf`
12. Nguyen, L.: Horn knowledge bases in regular description logics with PTime data complexity. Fundamenta Informaticae **104**(4), 349–384 (2010)
13. Nguyen, L.: A deterministic Horn fragment of CPDL$_{reg}$ with PTime data complexity. Manuscript, `http://www.mimuw.edu.pl/~nguyen/DHornCPDLreg.pdf` (2018)
14. Nguyen, L., Nguyen, T., Szałas, A.: Towards richer rule languages with polynomial data complexity for the Semantic Web. Data Knowl. Eng. **96**, 57–77 (2015)
15. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In: Proc. of IJCAI. pp. 1039–1044 (2011)
16. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Proceedings of IJCAI. pp. 466–471 (1991)

Deep Learning guinea pig image classification using Nvidia DIGITS and GoogLeNet

Lukasz Zmudzinski

University of Warmia and Mazury in Olsztyn, Poland
lukasz@zmudzinski.me

Abstract. In this paper guinea pig classification using deep learning imaging methods was performed on the Nvidia DIGITS 6. Models capable of distinguishing skinny, abyssinian and crested fur types were created in the process. To increase the classification accuracy empty images (with only the background) were added to the data set. Upon evaluation, the created model recognized the animals correctly from images taken in various household backgrounds.

Keywords: deep learning · animal recognition · robotics

1 Introduction

Robotic systems are nowadays increasingly appearing in various industries. This trend is also represented in various animal care facilities like farms, daries, shelters [1,2] and more. New robotic systems are created, that fill the public space as well as connect to the personal home environment. With the growing need of automating work more software and hardware platforms are employed to increase the ease of life.

In this work, deep learning techniques were utilized to create a classification model of guinea pigs in different home environments (living room, office, corridor, etc.), to explore the possibilities of bringing such systems into the world of household animals. Creating such a model was important, to understand how machine learning algorithms would adapt to live creatures, while keeping a high accuracy of the prediction and short inference times needed in robotics.

To address these problems GoogLeNet was used. The pre-trained model connects various techniques known from Deep Learning like convolutions, pooling, adding softmax and more [3], to distinguish all the objects that are present in the image. It implements so called *Inception modules*, that range from 245 filters to 1024 in top inception modules. The consequence of this is the possibility to remove fully connected layers on top completly [4].

Gathered results, will perform as a base for future projects of animal social and care systems. Example appliances could include:

- automatic feeding and cleaning systems,
- automatic pet door management,

– illness and status detectors,
– or mobile, home robots allowing the owners to check on their pets using mobile software.

1.1 Related Works

Related works about animal classification were published in regard of wild animal monitoring [5]. The authors used convolutional neural networks to create a model from the Serengeti National Park camera-trap database snapshot containing 179683 images. Then they tested the set using popular topologies like AlexNet, VGGNet, GoogLeNet and finally ResNets.

Similarly convolutional neural networks were used to recognize 20 species common in North America [6] over a 14346 image training data set. The imagery data from motion triggered cameras was automatically segmented using the graph-cut algorithm.

Another approach was taken for classifying different animals for automated species recognition [7]. The authors enforced ScSPM (Sparse coding Spatial Pyramid Matching) [8] to extract and classify animal species over a 7 thousand image data set. After that multi-class pre-trained SVMs were applied to classify global features of animal species.

All mentioned authors follow a similar pattern when using machine learning for image classification, but none of them concern household animals. All presented projects face different problems from those, that could be encountered in a safe, indoor environment. Hence, different data acquisition techniques had to be used.

1.2 Nvidia DIGITS

DIGITS (Nvidia Deep Learning GPU Training System) was used in this project. It is an open-source project for training deep neural networks (DNNs). The software simplifies common deep learning tasks such as managing data, designing and training neural networks and monitoring performance in real time. [9, 10].

The solution comes with pre-trained models (but it allows usage of self created ones) for example:

– GoogLeNet (Inception),
– AlexNet,
– UNET,
– and more.

In this paper GoogLeNet was used as the model of choice.

2 Background / Formulation

Deep Learning (DL) is a machine learning technique growing in popularity over the past few years. It is connected to the fact that it becomes more useful than before thanks to the amount of available training data and advances in computer hardware/software [11]. It allows to create models that perform the following tasks:

- computer vision,
- speech recognition,
- natural language processing,
- recommendation systems,
- and more.

To understand how DL works, we should first define what learning actually is. A simple definition was provided by Mitchell [12] as follows:

Definition 1. *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at task in T, as measured by P, improves with experience E.*

This can apply to different kind of tasks, performance measures and experiences. In this paper we will focus on the task of object classification using computer vision.

Classification of objects is based on describing to what category a given input belongs. This can be used in robotics for tasks like delivering foods and drinks to clients by the Willow Garage PR2 robot [13]. The general rule is to create an algorithm that produces the funcion: $f : \mathbb{R}^n \to \{1, ..., k\}$, where the category is assigned when $y = f(x)$ for input x.

This paper focused on supervised learning, which means that all data set items were associated with a label (each guinea pig image was added to a specific category). In practice, it means that the algorithm knew how to classify certain objects with similary properties from the start.

2.1 Machine learning and neural networks

Articial neural networks are a subgroup of algirthms that are used for machine learning. The main idea behind them is creating artificial neurons, wchich are implemented as a non-linear function over a linear combination of input features. Each neuron generally consist of one to multiple inputs with wages, activation function, bias and one output.

In neural network algorithms we can tweak several parameters, that will greately impact the final output: number of epochs, learning rate, solver type and many more.

- **Epoch** - one complete pass of the data in the data set. The amount of epochs should be determined through tests. Small number of epochs often leads to bad predictions, while a big number leads to overfitting.

– **Learning rate** - describes the rate at which the network abandons old beliefs, for new ones to take their place. The value must be correct, values that are too big or too small may lead to bad predictions.
– **Solver type** - contains information about how the weights are updated for the network. This project used Stochastic Gradient Descent (SGD) [14] and Adam.

When working with machine learning algorithms, we can encounter two problems, that are the result of our actions - underfitting and overfitting.

Overfitting takes place, when the model is training the data too well. That happens when noise, details or random fluctuations are taken into consideration, which negatively impacts the performance of the model on any new data. In such case, the parameters should be adjusted to constrain the amount of detail that the model learns.

Underfitting on the other hand is usually the result of big learning rates. The model becomes too general, which in the end gives bad results for object classification. To provide a solution to the problem, usually adjusting parameters or using different ML algorithms should be used.

2.2 Optimizers used in the project

Stochastic Gradient Descent (SGD) is one of the most popular algorithms in DL. It is an extension to the first-order optimization algorithm: Gradient Descent. In SGD, the gradient is an expectation, which may be estimated even using a small set of samples. SGD has proven to work very well with deep learning models. While it doesn't guarantee finding the local minimum, it usually finds a very low value of the cost funcion quickly.

The estimate from the example x minibatch m' can be written as follows: $g = \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} L\left(x^i, y^i, \theta\right)$, where the loss is $L\left(x, y, \theta\right) = -\log p\left(y|x; \theta\right)$ for each example.

Adam is an optimization algorithm that is used for iteratively updating the network weights based on the training data. The algorithm combines two extensions of the SGD: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation(RMSProp). Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).

2.3 Convolutional Networks

Convolutional Neural Networks (CNNs) are used for data, that has a known grid-like topology. The name comes from the fact, that it employs an operation called **convolution**, which is a kind of linear operation (instead of general matrix multiplication).

Convolutions are operations on two functions of a real value argument. The convolution can be represented as $s(t) = (x * w)(t)$, where x is a single input, w

is a valid probability density function and t is time. It leverages three important ideas that can improve machine learning systems: sparse interactions, parameter sharing and equivariant representations [11]. The output of the function is often called the feature map.

Each layer of CNNs consists of three distinguishable stages: producing sets of linear activations, detector stage and pooling. Pooling is a method that instead of giving the output of the neural net, provides a summary statistic of all nearby outputs. This is especially helpful, if images of variable size are given as an input. Moreover it helps with feature extraction (as it is known, neural networks loose data over time, with each layer) from convolutional layers.

2.4 GoogLeNet

GoogLeNet was introduced at ILSVRC 2014 competition, where it took first place with a result of 6.67% error rate (which is close to human level performance). The architecture consisted of 22 layers (27 with pooling) of the Deep CNN reducing the number of parameters to 4 million (60 million compared from AlexNet).

The main innovation between normal CNNs and GoogLeNet was the implementation of *Inception modules*. The modules ran several small convolutions in order to reduce the number of parameters. Moreover batch normalization, RMSprop and image distortions were used. Data from the previous layer was run over four 1x1 convolutions, one 3x3 convolution and one 5x5 convolution, with a 3x3 pooling added simultaneously. You can see the network presented on the graph on Fig. 1.

The Inception module 3x3 and 5x5 condolutions ratio increases as higher layers are achieved. This is due to the fact, that stacking Inception modules on top of each other produces an effect where as features of higher abstraction are captured by higher layers, their spatial concentration is expected to decrease [15].

The highest pro of the network is high inference speeds. GoogLeNet was designed to be computational efficient, so that it could be run on devices with limited computational power or low-memory footprint, making it a good choice for robotics and its applications.

2.5 Deep Learning downsides

While Deep Learning is performing well for many cases of image classification, it still has disadvantages, that should be considered when picking the right method. The most known cons of the method are:

- small data sets can produce bad results,
- long calculation time is a big factor,
- debugging is extremly hard,
- picking good neural net parameters takes practice,
- it's hard to gather the logic behind results.

With recent advantages in Deep Learning some of the factors are less limiting. The learning duration for example, is being decreased using parallellization.

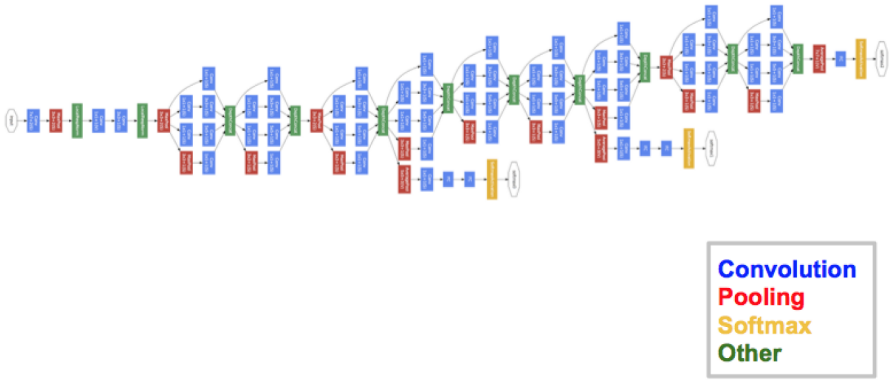


Fig. 1. GoogLeNet 22 Convolutional Neural Network architecture featuring inception layers.

3 Data Acquisition

The data was collected by recording a square video of each guinea pig over a period of 30 seconds in different environments and then extracting frames as an image. Each frame was then lowered in resolution to 256x256 px in order to fit GoogLeNet requirements. Example images taken from the data set can be seen in Fig. 2.

The entire data set consisted of 1098 images. From the initial data set - 25% (274) images were excluded for model validation and 10% (110) were excluded to calculate the test data loss and accuracy. Moreover 32 photos in different environments (animal cage, sleepingroom, guestroom, balcony and bathroom) were taken after the training to test the model behaviour. The visual representation of the training data set can be seen on Fig. 3.

To ensure good accuracy of the model, images for each guinea pig had to cover the whole anatomy of the animal. To achieve that, the following camera positions were covered:

- facing the mouth,
- both sides front and back facing,
- rear view of the animal,
- top view with different distances to the guinea pig.

Moreover empty images (without guinea pigs) were added with the same room backgrounds where previous photos were taken, to increase classification accuracy in distinguishing wanted objects. This set contained 191 images mixed with the guinea pig data set.

Guinea pigs phenotype highly depends on their breed. Therefore three different subjects of diverese ages were used in the experiment:

Fig. 2. Example images for crested, abyssinian and skinny guinea pigs from the provided data set.

– Fifi (4 years, abyssinian),
– Rey (2 years, crested),
– Asajj (2 years, skinny).

Using the data, four labels were produced and assigned to the following neural network classes, which provided a base for further classification:

– **None** - when there is no guinea pig in the image,
– **Abyssinian**, **Crested** and **Skinny** - fur type.

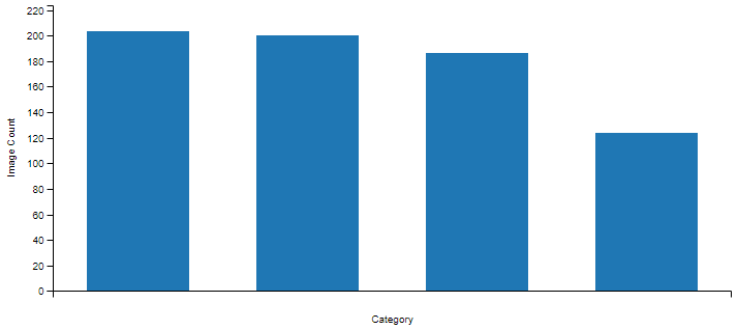


Fig. 3. Data representation in the training data set. Category labels from left: crested, skinny, abyssinian, none.

Finally the data was processed by the neural network, on different settings. Two parameters were changed during testing: learning rate and optimizer used, while the epoch count remained at 15. The settings can be seen below:

1. **First test:**
   - Epoch count: 15,
   - Learning rate: 0.001 with fixed policy,
   - Optimizer: Stochastic Gradient Descent.
2. **Second test:**
   - Epoch count: 15,
   - Learning rate: 0.001 with step down policy,
   - Optimizer: Adam.
3. **Third test:**
   - Epoch count: 15,
   - Learning rate: 0.01 with step down policy,
   - Optimizer: Stochastic Gradient Descent.

4 Results

Final results for classification models were gathered from the neural networks described in the previous section.

Performing the first test gave good results, although overfitting was discovered around epoch 13, with accuracy of 98,95% and loss of 0.04. Later epochs drastically fell in value, producing an accuracy of 65,63% and loss of 1.03. Overfitting appeared because of the low learning rate from the very start, which should have been avoided.

Second test, using Adam as the solver type, provided the best results. The final acurracy and loss after 15 epochs were 99.31% and 0.04 respectfully. A different training rate (0.01) was also tested, but it didn't provide any useful results.

The final test gave good results, but not satisfactory - it produced an accuracy of 87,15% and loss of 0.32. That was not enough to be used for the guinea pig classification system (the predictions would give false-positives).

The final classification model that was selected for further use and testing, was taken from test number two, using the Adam optimizer. Over 15 epochs with a learning rate of 0.001 ran on the GoogLeNet model on Nvidia DIGITS, it has provided the best results. Using more epochs and a different learning rate was tested afterwards, but it led to overfitting of the data. You can see the final result on Fig. 4.

4.1 Manual image test results

After the model was created, a series of tests were performed to check, if the inference is correct. The first set of tests consisted of images from the previously gathered data set. The prediction was always right for provided images, with the value between 70%-95%. You can see an example result on Fig. 5.

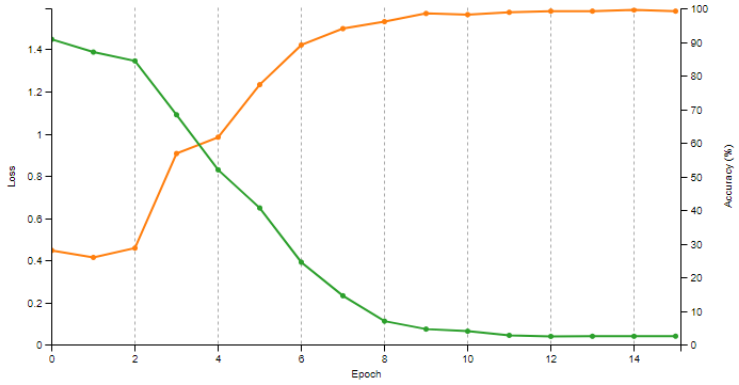


Fig. 4. Graph showing the final accuracy (orange) and loss (green) of the second test running Adam optimizer.

Moreover, as soon as acquiring enough information from predictions was done, tests on new images were performed (different environments, same guinea pigs) giving satisfactory results - guinea pigs were classified correctly on each provided picture containing the animal. Example result one such case can be seen on Fig. 6.

One failed prediction was encountered, when a picture of a background with a cat was used. The cat was badly classified as an abyssinian guinea pig. This happened due to the fact, that the data set didn't contain images that were in any case similar to the cat picture used. More cases like that can be produced with the provided model, as it was trained on specific data in the first place.

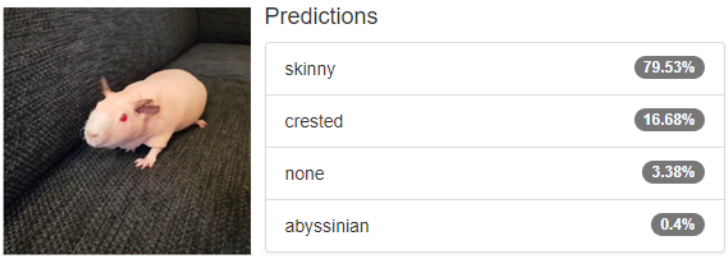


Fig. 5. Example correct prediction results (79.53%) using previously captured images in selected environments.

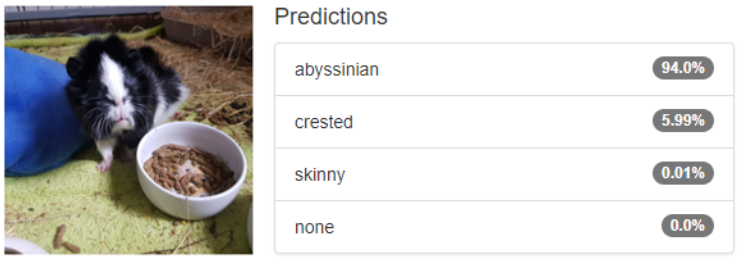


Fig. 6. Example correct prediction results (94.0%) using post-model captured images in animal cage environment.

5 Conclusion / Future work

Modern robotics highly depend on sensor readings of the surrounding environment. They often use camera input as one of the parameters to perceive the world. Due to that, imaging methods for decision-making were introduced.

In this paper Deep Learning was implemented for guinea pig classification in order to explore the possibilities of introducing household animal care using robotics and automation, while keeping them safe.

The provided GoogLeNet model from Nvidia DIGITS has proven successful in identifying the guinea pigs in different environments, even when taking images that weren't originally added to the data set. Some errors were observed (false-positives) when no guinea pigs were present in the tested image. The model will behave poorly when other animals are present in the pictures, since the classification was based purely on guinea pigs.

Increasing the accuracy of the model, can greately improve the robot-animal interactions, allowing to tailor behaviours to specific beings. This could be achieved by using modifying the learning rate, using more images, creating more labels or finally using a different optimizer or pre-built model. There are many possible variables to take into consideration, when building a model for a defined task.

The guinea pig classification model after building with GoogLeNet, was able to provide results almost instantly. This is important, if used for robotics, because while waiting for an action, the environment can change quite drasticly. Moreover, such model are built to be deployed on an autonomous platform (Raspberry, Jetson TX2 or any other), so the memory usage will be limited, increasing the inference calculation time.

The created model proves that guinea pig fur recognition for robotic systems is possible. The project gave good results - the created model recognized the animals correctly from images taken in various household backgrounds. The prediction was acquired fast making the inference time low. This is especially important for robotic systems that deal with live animals, because the reaction times need to be rapid.

Future work might include robotic systems that monitor the state of specific animals, adjust food distribution depending on image readings or alert when the guinea pig suffers from any kind of illness.

Moreover, different types of models can be employed to see, which one fits the needs the most. The project shouldn't be limited to GoogLeNet.

References

1. J. J. Roldán, J. del Cerro, D. Garzón-Ramos, P. Garcia-Aunon, M. Garzón, J. de León, and A. Barrientos, "Robots in agriculture: State of art and practical experiences," *Service Robots Antonio Neves, IntechOpen*, 2018. DOI: 10.5772/intechopen.69874. Available from: https://www.intechopen.com/books/service-robots/robots-in-agriculture-state-of-art-and-practical-experiences.
2. M. BD, S. Adil, and S. Ranvir, "Robotics: An emerging technology in dairy and food industry: Review," *International Journal of Chemical Studies*, 2018.
3. J. Long, E. Shelhammer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *arXiv: 1411.4038*, (v2) 2015.
4. F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2017.
5. A. Gomez, A. Salazar, and F. Vargas, "Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks," 2016.
6. G. Chen, T. X. Han, Z. He, R. Kays, and T. Forrester, "Deep convolutional neural network based species recognition for wild animal monitoring," *International Conference on Image Processing (ICIP) IEEE pp. 858-862*, 2014.
7. X. Yu, J. Wang, R. Kays, P. A. Jansen, T. Wang, and T. Huang, "Automated identification of animal species in camera trap images," *EURASIP Journal of Image and Video Processing*, 2013.
8. J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
9. "Nvidia DIGITS." https://developer.nvidia.com/digits.
10. B. Erickson, P. Korfiatis, Z. Akkus, T. Kline, and K. Philbrick, "Toolkits and libraries for deep learning," *Journal of Digital Imaging vol. 30 pp. 400-405*, 2017.
11. I. Goodfellow and Y. Bengio, *Deep Learning*. Cambridge, Massachusetts: The MIT Press, 2016.
12. T. M. Mitchell, *Machine Learning*. McGraw-Hill, New York, 1997.
13. I. Goodfellow, N. Koenig, N. Muja, C. Pantofaru, and A. Sorokin, "Help me help you: Interfaces for personal robots," 2010.
14. L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proceedings of COMPSTAT 2010*.
15. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Deeper with convolutions," Available from: http://arxiv.org/abs/1409.4842.

A Novel Ensemble Model - The Random Granular Reflections

Piotr Artiemjew, Krzysztof Ropiak

Faculty of Mathematics and Computer Science
University of Warmia and Mazury in Olsztyn
Poland
email:artem@matman.uwm.edu.pl, kropiak@matman.uwm.edu.pl

Abstract. One of the most significant achievements in machine learning is the development of Ensemble techniques, which gave a powerful tool for tuning classifiers. The most popular methods are Random Forests, Bagging and Boosting. In this paper we present a novel ensemble model, named Random Granular Reflections. This algorithm creates an ensemble of homogenous granular decision systems. In each iteration of learning process, the training decision system is covered by random homogenous granules and the granular reflection is created, which takes part in classification process. Seeing the initial results - our approach is promising and seems to be comparable with the selected popular models.

Keywords: Random Granular Reflections, Homogenous Granulation, CSG Classifier, Ensemble Model, Rough Sets, Decision Systems, Classification

1 Introduction

This paper is about the application of granular rough computing in new Ensemble model. The techique that we use to prepare the data for each iteration of learning process was inspired by Polkowski standard granulation - see [16]. This method was the beginning of many new algorithms with diverse applications, for instance in Artiemjew [1]-[3], Polkowski [15]–[20], Polkowski and Artiemjew [21] we have the presentation of standard granulation, concept dependent and layered granulation in the context of training data size reduction, missing values absorbtion and usage in the classification processes.

In our recent works - see [24] and [25] - we have developed a new granulation technique - homogenous granulation (see. detail decription and toy example in Sect. 2). This approximation technique is based on creation of groups of r-indiscernible objects around each training object by lowering the ratio of indiscernibility until the granules contain only homogoneus objects in the sense of their decision class. In this method - what distinguishes it from previously studied - there is no need to estimate optimal parameter of approximation. The r-indiscernibility level for each central training object is formed in automatic way and depends on the homogeneity in decision classes.

The ensemble scheme of classification is really effective in many contexts, for instance in rough set methods the exemplary succesfull applications can be found in [6–8, 26, 29]. The recently developed approximation technique - homogenous granulation - gave us motivation to check it in ensemble model creation. Additionally to Random Forests, Bagging and Boosting we propose a novel algorithm - Ensemble of Random Granular Reflections. The method is based on representation of original training system by its granular reflections formed from random homogenous granules, which covers it in each iteration of learning process. Each granular reflection of training decision system is additionally reduced in size in comparison with original training decision system. The granular reflection of each iteration represents the internal knowledge from original system using the random coverage. The level of data reduction is up to 50 per cent of original data.

In this work we have first sight into this method and for simplicity we treat all attributes as cathegorical. For experiments we performed 50 iterations of learning process with use of CSG classifier - the classifier based on simple granules of knowledge - see [4].

We have compared our new method with selected ensemble models - see Sect. 3.

The rest of the work contains the following content. In Sect. 2 we have introduction to homogenous granulation algorithm. In Sect 3 we have brief introduction to selected Ensemble models. In Sect. 4 we present our novel ensemblme model - The Random Granular Reflections technique. In Sect. 5 we show the results of the experiments, and we conclude the paper in Sect. 6.

2 Homogenous granulation

Detailed theoretical introduction to rough inclusions is available in Polkowski [15] – [20].

For given objects u, v from training decision system, r granulation radius, and A the set of attributes, the standard rough inclusion μ is defined as

$$\mu(v, u, r) \Leftrightarrow \frac{|IND(u, v)|}{|A|} \geq r \tag{1}$$

where

$$IND(u, v) = \{a \in A : a(u) = a(v)\}, \tag{2}$$

The homogenous granules are formed as follows,

$$g_{r_u}^{homogenous} = \{v \in U : |g_{r_u}^{cd}| - |g_{r_u}| == 0,\ for\ minimal\ r_u\ fulfills\ the\ equation\}$$

where

$$g_{r_u}^{cd} = \{v \in U : \frac{IND(u, v)}{|A|} \leq r_u\ AND\ d(u) == d(v)\}$$

and

$$g_{r_u} = \{v \in U : \frac{IND(u,v)}{|A|} \leq r_u\}$$

$$r_u = \{\frac{0}{|A|}, \frac{1}{|A|}, ..., \frac{|A|}{|A|}\}$$

2.1 The process of training system covering

In the process of covering - the objects from training system are covered based on chosen strategy. We use simple random choice because it is the most effective method among studied ones - see [21]).

The last step of the granulation process is shown in the next section.

2.2 Granular reflections

In this step we formed the granular reflections of the original training system based on the granules from the found coverage (the coverage is the set of granules, which cover the universe of traning objects completly). Each granule $g \in COV(U, \mu, r)$ from the coverage is finally represented by single object formed using the Majority Voting (MV) strategy (choice the most common values).

$$\{MV(\{a(u) : u \in g\}) : a \in A \cup \{d\}\} \tag{3}$$

The granular reflection of the decision system $D = (U, A, d)$ is the decision system ($COV(U, \mu, r)$, the set of objects formed from granules.

$$v \in g_r^{cd}(u) \text{ if and only if } \mu(v, u, r) \text{ and } (d(u) = d(v)) \tag{4}$$

for a given rough (weak) inclusion μ.

Toy example of described granulation method is presented in the next section.

2.3 Toy example of homogenous granulation

Considering training decision system from Tab. 1.

Homogenous granules for all training objects:

$$g_0.75(u_1) = (u_1)$$

$$g_1(u_2) = (u_2)$$

$$g_1(u_3) = (u_3)$$

$$g_1(u_4) = (u_4)$$

Table 1. Training data system (U_{trn}, A, d), (a sample from Quinlan data set [23])

	a_1	a_2	a_3	a_4	d
u_1	*sunny*	*hot*	*high*	*strong*	*no*
u_2	*rain*	*cool*	*normal*	*strong*	*no*
u_3	*overcast*	*cool*	*normal*	*strong*	*yes*
u_4	*sunny*	*mild*	*high*	*weak*	*no*
u_5	*sunny*	*cool*	*normal*	*weak*	*yes*
u_6	*rain*	*mild*	*normal*	*weak*	*yes*
u_7	*overcast*	*hot*	*high*	*weak*	*yes*
u_8	*sunny*	*mild*	*normal*	*strong*	*yes*
u_9	*overcast*	*mild*	*high*	*strong*	*yes*
u_{10}	*rain*	*mild*	*high*	*weak*	*yes*
u_{11}	*overcast*	*hot*	*normal*	*weak*	*yes*

$$g_0.75(u_5) = (u_5)$$
$$g_0.75(u_6) = (u_6, u_{10})$$
$$g_0.75(u_7) = (u_7, u_{11})$$
$$g_0.75(u_8) = (u_8)$$
$$g_0.75(u_9) = (u_9)$$
$$g_1(u_{10}) = (u_{10})$$
$$g_0.5(u_{11}) = (u_3, u_5, u_6, u_7, u_{11})$$

Granules covering training system by random choice:

$$g_0.75(u_1) = (u_1)$$
$$g_1(u_2) = (u_2)$$
$$g_1(u_4) = (u_4)$$
$$g_0.75(u_6) = (u_6, u_{10})$$
$$g_0.75(u_7) = (u_7, u_{11})$$
$$g_0.75(u_8) = (u_8)$$
$$g_0.75(u_9) = (u_9)$$
$$g_0.5(u_{11}) = (u_3, u_5, u_6, u_7, u_{11})$$

Granular decision system from above granules is as follows:

Table 2. Granular decision system formed from Covering granules

$g_0.75(u_1)$	*sunny*	*hot*	*high*	*strong*	*no*
$g_1(u_2)$	*rain*	*cool*	*normal*	*strong*	*no*
$g_1(u_4)$	*sunny*	*mild*	*high*	*weak*	*no*
$g_0.75(u_6)$	*rain*	*mild*	*normal*	*weak*	*yes*
$g_0.75(u_7)$	*overcast*	*hot*	*high*	*weak*	*yes*
$g_0.75(u_8)$	*sunny*	*mild*	*normal*	*strong*	*yes*
$g_0.75(u_9)$	*overcast*	*mild*	*high*	*strong*	*yes*
$g_0.5(u_{11})$	*overcast*	*cool*	*normal*	*weak*	*yes*

In the next section there is a brief description of the selected popular Ensemble models.

3 Selected popular ensemble models

There are many techniques in the family of Enssemble models. One of the most popular are Random Forests, Bagging and Boosting - see [31]. Short description of mentioned models is to be found below.

Bootstrap Ensembles - Pure Bagging: It is the random committee of bootstraps [33]. It is a method in which the original decision system - the basic knowledge - is split into (TRN) training data set, and ($TSTvalid$) validation test data set. And from the TRN system, for a fixed number of iterations, we form a new Training systems ($NewTRN$) by random choice with returning of $card\{TRN\}$ objects. In all iterations we classify the TRNvalid system in two ways: the first based on the actual $NewTRN$ system and the second based on the committee of all performed classifications. In the committee majority voting is performed and the ties are resolved randomly.

Bagging based on Arcing - Bagging: The main difference between this method and Bootstrap Ensembles is that the TRN is split into two data sets $NewTRN$ and $NewTST$ - see [5] and [27]. This split is based on Bootstraps where weights determine the probability with which objects are assigned to NewTRN set. Initially weights are equal, but after first classification of the NewTST using NewTRN weights are lowered for well-classified objects. The next step is normalization of weights. This algorithm which shows forming of Bootstraps is called Arcing. Classifying the $TSTvalid$ with $NewTRN$ in a single iteration as the committee of classifiers is the last step of this method. In Arcing weights are modified with the factor equal to $\frac{1-Accuracy}{Accuracy}$.

Boosting based on Ada-Boost with Monte Carlo split: Classification method used in this algorithm is similar to the previously described with the difference that the NewTRN and NewTST are formed in a different way - see [9], [28] and [34].

Objects for NewTRN are chosen based on weights and fixed ratio is used to split the TRN data set. Previous experiments show that split ratio equal to 0.6 is optimal, as it is close to the approximate size of the distinguishable objects in the bootstraps. Other parts of this algorithm works like in the previous one.

Random forests: In this model random trees are created based on randomly chosen attributes and then they take part in the classification process in each iteration. This method can be usefull in other classifiers using the random set of attributes before usage in classification process. The number of attributes, which should be chosen depending on internal data logic, have to be found in an experimental way.

In the following section we present introduction to our new Ensemble method.

4 Ensemble of Random Granular Reflections

In each iteration of our new ensemble model we have used a different homogenous granular decision system formed from random homogenous granules, which covers the original training system. The visualization of the model can be found in Fig. 1.

The time comlexity of this model is quadratic. The most time consuming part is granulation, which main component takes $((no.\_of\_obj.)^2) * (no.\_of\_att.)$ operations.

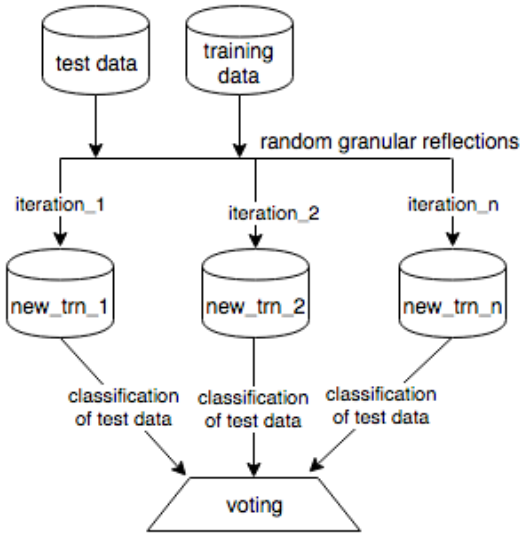


Fig. 1. Ensemble of Random Granular Reflections

5 Experimental Session

To perform initial experiments we used the australian credit data set from UCI Machine Learning Repository [30]. We have run our algorithm with 50 iterations of learning for each tested Ensemble model. As a reference point we have chosen Committee of Bootstraps (Pure Bagging) [33], Boosting based on Arcing (Bagging) [5], [27], and Ada-Boost with Monte Carlo split [9], [28] and [34] - for details see Sect. 3. As a reference classifier we used CSG classifier [4] with radius 0.5. The effectiveness is evaluated by percentage of properly classified objects - the accuracy.

The first result of Random Granular Reflections technique for chosen data set is presented in Fig. 2. The results of the other popular ensemble models are to be found in Figs. 3, 4 and 5. For selected data set our new technique outperformed the other checked methods.

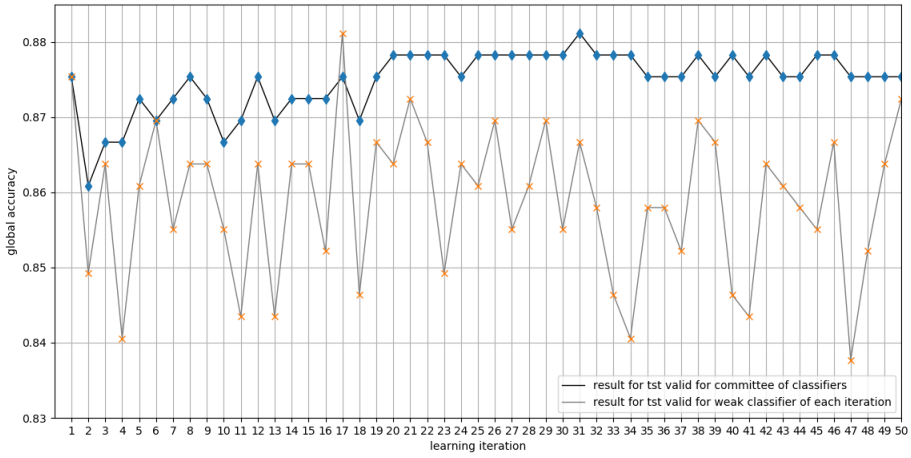


Fig. 2. Ensemble of Random Granular Reflections for australian credit dataset - the accuracy of classification - 5 times 50 iterations of learning

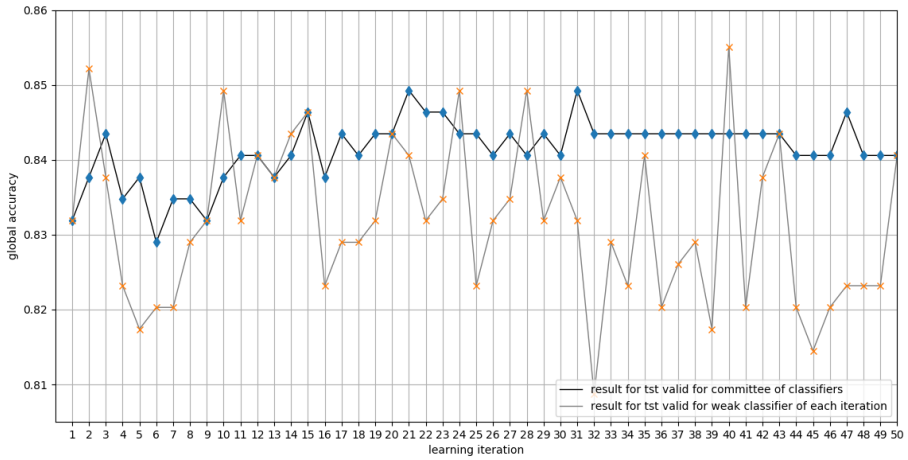


Fig. 3. Bagging ensemble model for australian credit dataset - the accuracy of classification - 5 times 50 iterations of learning

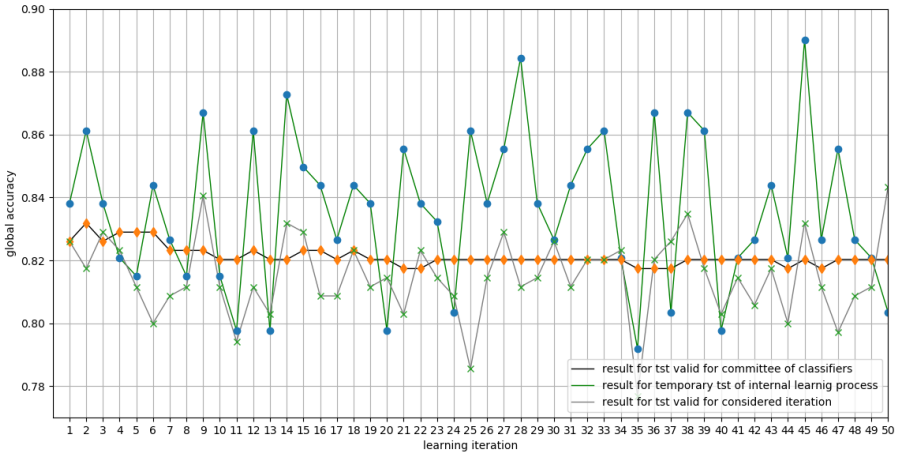


Fig. 4. AdaBoost ensemble model for australian credit dataset - the accuracy of classification - 5 times 50 iterations of learning

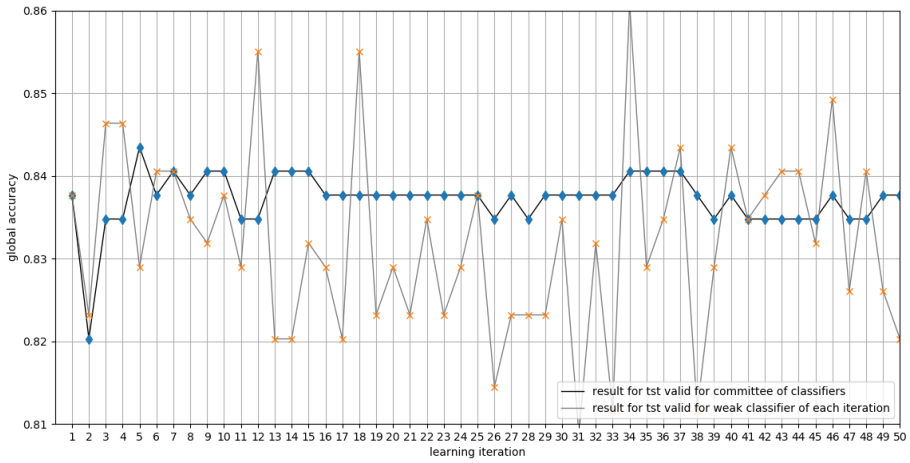


Fig. 5. Pure Bagging ensemble model for australian credit dataset - the accuracy of classification - 5 times 50 iterations of learning

6 Conclusions

The results of the experiments show the effectivenes of our new technique. The Ensemble of Random Granular Reflections turn out to be competitive with other techniqes like Bagging and Boosting. Despite promising initial results, much is left to be done to evaluate the effectiveness and set of applications of this new method.

In the future works we have a plan to extensively check the effectiveness of new model and we are planning to apply the other types of granules in the proposed ensemble model.

7 Acknowledgements

The research has been supported by grant 23.610.007-300 from Ministry of Science and Higher Education of the Republic of Poland.

References

1. Artiemjew, P. : Classifiers from Granulated Data Sets: Concept Dependent and Layered Granulation. In Proceedings RSKD'07. The Workshops at ECML/PKDD'07, Warsaw Univ. Press, Warsaw, 2007, pp 1–9 (2007)
2. Artiemjew, P.: Rough mereological classifiers obtained from weak rough set inclusions. In Proceedings of Int. Conference on Rough Set and Knowledge Technology RSKT'08, Chengdu China, Lecture Notes in Artificial Intelligence, vol. 5009. Springer Verlag, Berlin, 2008, pp 229–236 (2008)

3. Artiemjew, P. (2013): A Review of the Knowledge Granulation Methods: Discrete vs. Continuous Algorithms. In Skowron A., Suraj Z. (eds.)(2013): Rough Sets and Intelligent Systems. ISRL 43, Springer-Verlag, Berlin, 2013, pp 41–59.
4. Artiemjew, P.: Boosting Effect of Classifier Based on Simple Granules of Knowledge, In: Information technolojy and control, Print ISSN: 1392-124X, Vol 47, No 2 (2018)
5. Breiman, L.: Arcing classifier (with discussion and a rejoinder by the author). Ann. Statist. 26 (3): 801849. Retrieved 18 January 2015. Schapire (1990) proved that boosting is possible. (Page 823) (1998)
6. Hu, X., Construction of An Ensemble of Classifiers based on Rough Sets Theory and Database Operations, Proc. of the IEEE International Conference on Data Mining (ICDM2001), (2001)
7. Hu, X.: Ensembles of classifiers based on rough sets theory and set-oriented database operations, Presented at the 2006 IEEE International Conference on Granular Computing, Atlanta, GA (2006)
8. Murthy, C., Saha, S., Pal, S.K.: Rough Set Based Ensemble Classifier, In: Rough Sets, Fuzzy Sets, Data Mining and Granular Computing Lecture Notes in Computer Science Volume 6743, p. 27 (2001)
9. Ohno-Machado, L.: Cross-validation and Bootstrap Ensembles, Bagging, Boosting, Harvard-MIT Division of Health Sciences and Technology, http://ocw.mit.edu/courses/health-sciences-and-technology/hst-951j-medical-decision-support-fall-2005/lecture-notes/hst951_6.pdf HST.951J: Medical Decision Support, Fall (2005)
10. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11, pp 341–356 (1982)
11. Polkowski, L.: Rough Sets. Mathematical Foundations. Physica Verlag, Heidelberg (2002)
12. Polkowski, L.: A rough set paradigm for unifying rough set theory and fuzzy set theory. Fundamenta Informaticae 54, pp 67–88; and : In Proceedings RSFDGrC03, Chongqing, China, 2003. Lecture Notes in Artificial Intelligencevol. 2639, Springer Verlag, Berlin, pp 70–78 (2003)
13. Polkowski, L.: Toward rough set foundations. Mereological approach. In Proceedings RSCTC04, Uppsala, Sweden. Lecture Notes in Artificial Intelligence vol. 3066, Springer Verlag, Berlin, pp 8–25 (2004)
14. Polkowski, L.: Granulation of knowledge in decision systems: The approach based on rough inclusions. The method and its applications In Proceedings RSEISP'07, Lecture Notes in Artificial Intelligence vol. 4585. Springer Verlag, Berlin, pp 69– (2004)
15. Polkowski, L.: Formal granular calculi based on rough inclusions. In Proceedings of IEEE 2005 Conference on Granular Computing GrC05, Beijing, China. IEEE Press, pp 57–62 (2005)
16. Polkowski, L.: A model of granular computing with applications. In Proceedings of IEEE 2006 Conference on Granular Computing GrC06, Atlanta, USA. IEEE Press, pp 9–16 (2006)
17. Polkowski, L.: The paradigm of granular rough computing. In Proceedings ICCI'07, Lake Tahoe NV. IEEE Computer Society, Los Alamitos CA, pp 145–163 (2007)
18. Polkowski, L.: A Unified Approach to Granulation of Knowledge and Granular Computing Based on Rough Mereology: A Survey, in: Handbook of Granular Computing, Witold Pedrycz, Andrzej Skowron, Vladik Kreinovich (Eds.), John Wiley & Sons, New York, 375-401 (2008)

19. Polkowski, L.: Granulation of Knowledge: Similarity Based Approach in Information and Decision Systems. In Meyers, R. A.(ed.): Encyclopedia of Complexity and System Sciences. Springer Verlag, Berlin, article 00788 (2009)
20. Polkowski, L.: Approximate Reasoning by Parts. An Introduction to Rough Mereology. Springer Verlag, Berlin, (2011)
21. Polkowski, L., Artiemjew, P.: Granular Computing in Decision Approximation - An Application of Rough Mereology, in: Intelligent Systems Reference Library 77, Springer, ISBN 978-3-319-12879-5, 1-422 (2015)
22. Poap, D., Woniak, M., Wei, W., Damaeviius, R.: Multi-threaded learning control mechanism for neural networks. Future Generation Computer Systems, Elsevier 2018.
23. Quinlan, J., R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, Kluwer Academic Publishers (1993)
24. Ropiak, K., Artiemjew, P.: On Granular Rough Computing: epsilon homogenous granulation, In: Proceedings of International Joint Conference on Rough Sets, IJCRS'18, Quy Nhon, Vietnam, Lecture Notes in Computer Science (LNCS), vol. 11103, Springer, Heidelberg, pp 546–558 (2018)
25. Ropiak, K., Artiemjew, P.: A Study in Granular Computing: homogenous granulation. In: Dregvaite G., Damasevicius R. (eds) Information and Software Technologies. ICIST 2018. Communications in Computer and Information Science, Springer (2018) in print
26. Saha, S., Murthy, C.A., Pal, S.K.: Rough set based ensemble classifier for web page classification. Fundamenta Informaticae 76(1-2), 171187 (2007)
27. Schapire, R.E.: A Short Introduction to Boosting (1999)
28. Schapire, R.E.: The Boosting Approach to Machine Learning: An Overview, MSRI (Mathematical Sciences Research Institute) Workshop on Nonlinear Estimation and Classification (2003)
29. Shi, L., Weng, M., Ma, X., Xi, L.: Rough Set Based Decision Tree Ensemble Algorithm for Text Classification, In: Journal of Computational Information Systems6:1, 89-95 (2010)
30. University of California, Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/index.php
31. Yang, P., Yang, Y., H., Zhou, B., B.; Zomaya, A., Y.: A review of ensemble methods in bioinformatics: Including stability of feature selection and ensemble feature selection methods. In Current Bioinformatics, 5, (4):296-308, 2010 (updated on 28 Sep. 2016)
32. Zadeh, L. A.: Fuzzy sets and information granularity. In Gupta, M., Ragade, R., Yager, R.R. (eds.): Advances in Fuzzy Set Theory and Applications. North–Holland, Amsterdam, 1979, pp 3–18 (1979)
33. Zhou, Z.-H.: Ensemble Methods: Foundations and Algorithms. Chapman and Hall/CRC. p. 23. ISBN 978-1439830031. The term boosting refers to a family of algorithms that are able to convert weak learners to strong learners (2012)
34. Zhou, Z.-H.: Boosting 25 years, CCL 2014 Keynote (2014)

The Hierarchical Learning Algorithm for Deep Neural Networks

Stanislaw Placzek[1] and Aleksander Placzek[2]

[1] University of Business in Wroclaw, Poland
stanislaw.placzek@wp.pl
[2] Silesian University of Technology, Faculty of Automatic Control, Electronic and Computer Science, WASKO Gliwice, Poland
a.placzek@wasko.pl

Abstract

In the article, the emphasis is put on the modern artificial neural network (ANN) structure, which in the literature is known as a deep neural network. A network includes more than one hidden layer and comprises many standard modules with ReLu nonlinear activation function. A learning algorithm includes two standard steps, forward and backward, and its effectiveness depends on the way the learning error is transported back through all the layers to the first layer. Taking into account all the dimensionalities of matrixes and the nonlinear characteristics of ReLu activation function, the problem is very challenging. In practice tasks, a neural networks internal layer matrixes with ReLu activations function, include a lot of null value of weight coefficients. This phenomenon has a negative impact on the effectiveness of the learning algorithm's convergence. Analyzing and describing an ANN structure, one usually finds that the first parameter is the number of ANNs layers "L". By implementing the hierarchical structure to the learning algorithm, an ANN structure is divided into sub-networks. Every sub-network is responsible for finding the optimal value of its weight coefficients using a local target function to minimize the learning error. The second coordination level of the learning algorithm is responsible for coordinating the local solutions and finding the minimum of the global target function. In each iteration the coordinator has to send coordination parameters into the first level of subnetworks. By using the input and the teaching vectors, the local procedures are working and finding their weight coefficients. At the same step the feedback error is calculated and sent to the coordinator. The process is being repeated until the minimum of all the target functions is achieved.

1 Deep neural network structure

A deep neural network is built with topologically and logically uniform modules known as layers. A networks structure includes an input layer, a lot of hidden layers, and finally an output layer. Usually, a network is built with $1 \div L$ layers of different or identical structure. From a mathematical point of view, a layer could be described by a matrix of weight coefficients W^l, an input vector $X^{(l-1)}$, a hidden vector U^l, an activation function $ReLu(U) = max(0, U)$, and an output vector X^l (Fig.1).

A deep neural network includes $L \div 1$ hidden layers and one output layer which contains a different activation function. An output layer needs to aggregate a set of partial features from previous layers to achieve the final result, that is an output signal. Using Fig.1, one can define the target function:

$$\Phi = \frac{1}{2} \cdot (X^L - Y)^T \cdot (X^L - Y) \tag{1}$$

Where:
$X^L[1 \div N^L]$-the output vector,

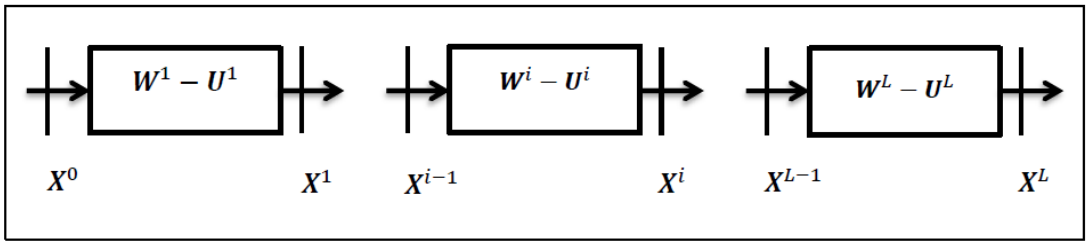


Figure 1: Deep Neural Network structure

N^L-the dimensionality of the output vector,
$Y[1 : N^L]$-the vector of teaching data.

For all the hidden layers and for forward calculation, one can write:

$$U^l = W^l \cdot X^{l-1} \tag{2}$$

$$X^l = F(U^l) \tag{3}$$

Where:
$l = 1 \div L$- number of layers in a deep neural network,
U^l- the internal vector for layer l,
F - the vector of activation function,
W^l-the matrix of weight coefficients for layer l,
X^{l-1}, X^l - the input and output vector of layer l, accordingly.

The process of selecting an activation function is an essential and difficult task. When building standard networks, usually sigmoid and tanh activation functions are used. A sigmoid function has two areas of value in which the function, in an asymptotic way, achieves the value of zero or one. This characteristic has a negative impact on the derivative value and, at the same time, on the algorithm convergence. At the moment, a new activation function is used in a deep neural network, a Rectified Linear Unit: $ReLu$, which is defined as follows:

$$f = ReLu(u) = max(0, u) \tag{4}$$

From a mathematical point of view, this function is discontinuous for $u = 0$. In a computers application, this problem is solved by the accepted value $f = 0$ for $u = 0$.

1.1 Learning algorithm

In computer applications, a back propagation learning algorithm is the most popular one. A learning error is calculated from an output layer, through all the hidden layers to the input data. From (1), one can calculate the first derivatives for the output layer, denoting:

$$\frac{\partial \Phi}{\partial X^L} = E^L = X^L - Y \tag{5}$$

$$\frac{\partial \Phi}{\partial W^L} = \frac{\partial \Phi}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial W^L} = E^L \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial W^L} \tag{6}$$

Where:
$E^L = [\epsilon_1^L, \epsilon_2^L, ...\epsilon_{N^L}^L]$-the vector of an output layer error.

The derivatives of the ReLu function could be written as follows:

$$\frac{\partial X^L}{\partial U^L} = max(0, U^L)' = 1(U^L) \tag{7}$$

The last part of equation (6) is calculated:

$$\frac{\partial U^L}{\partial W^L} = X^{L-1} \tag{8}$$

Finally, formula (6) can be written in the matrix form using the Hadamard $\odot$ product notation :

$$\frac{\partial \Phi}{\partial W^L} = \{E^L \odot 1(U)^L\} \cdot (X^{L-1})^T \tag{9}$$

$$\frac{\partial \Phi}{\partial X^{L-1}} = \frac{\partial \Phi}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial X^{L-1}} = E^L \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial X^{L-1}} \tag{10}$$

Using the same notation for an output layer, formula (10) can be written as:

$$E^{L-1} = \frac{\partial \Phi}{\partial X^{L-1}} = E^L \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial X^{L-1}} \tag{11}$$

In the matrix form:

$$E^{L-1} = (W^L)^T \cdot \{1(U)^L \odot E^L\} \tag{12}$$

The output layer error E^L has to be translated into the previous layer $L-1$, this process will be repeated up to the first hidden layer. Fig. 3. shows the full scheme for a back propagation algorithm and how a layer's error is translated back through the network. The final back propagation formulas have a recurrent structure. An algorithm will start from the output layer L and going through all the hidden layers, will achieve the first hidden layer 1:

$$E^{l-1} = (W^l)^T \cdot \{1(U)^l \odot E^l\} \tag{13}$$

$$\frac{\partial \Phi}{\partial W^l} = \{E^l \odot 1(U)^l\} \cdot (X^{l-1})^T \tag{14}$$

Where:
$l = 1 : L$.
According to Fig. 2. the layer is laid between input vector $X^{(l-1)}$ and output vector X^l. The same border is used for the back propagation error from E^l to $E^{(l-1)}$. In a standard neural network, a sigmoid activation function is used. An output error is translated back to the first layer decreasing its value and finally, an algorithm can calculate zero's value. This property has a very negative impact on convergence. Therefor, it is the main reason to use *ReLu* activation function, especially in a deep neural network. Its derivative is equal to 1 or to 0 - the Heaviside step function. Some derivatives (14) are equal to 0 and the weight coefficient does not change in the actual iteration process (formula 14). The same can be observed in the error back propagation function (formula 13).

Taking into account all the limitations mentioned above, a neural network learning algorithm is decomposed, and a new coordination level is implemented. Two or more levels could be used to improve the learning algorithm convergence.

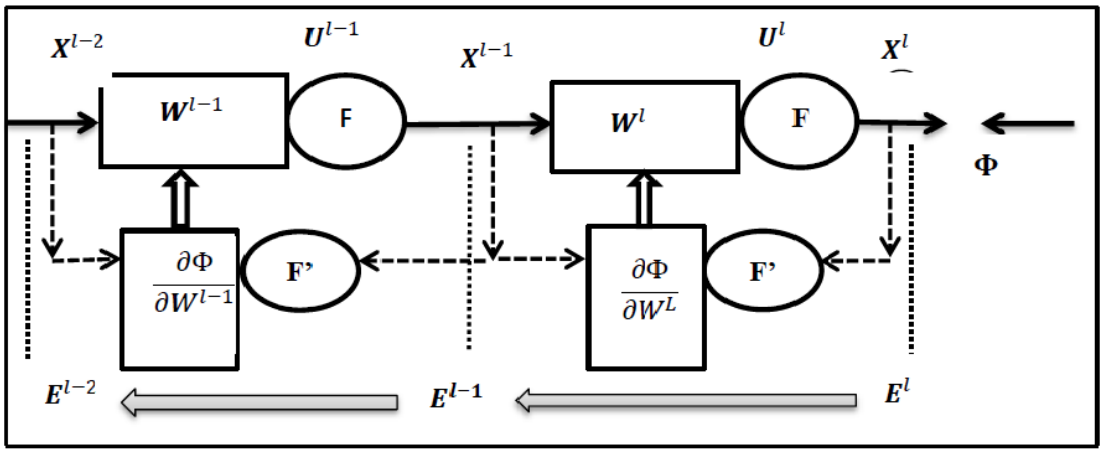


Figure 2: A scheme of an error back propagation through the layers

2 Learning algorithm decomposition

For a multi-layer ANN, a lot of hidden layers and one output layer are sectioned off. The smaller part will be described as a sub-network. Every sub-network has its own output vector, an input vector of the succeeding one X^l, and a local target function Φ^l where $l = 1 \div L$. Because of the specific organization of an ANNs hierarchy there are many sub-networks on the first level, for each of which local target functions are defined:

$$\Phi = \{\Phi^1, \Phi^2 .... \Phi^l ... \Phi^{L-1}, \Phi^L\} \tag{15}$$

These sets of local tasks have to be coordinated to achieve the global solution. The coordinator, as an independent task, will have its own target function. Taking everything into account, this concept is the base on which one may build the new scheme of the ANN learning algorithm's structure (Fig. 3). It is the neural network 's and the learning algorithm's hierarchical structure. The two-level ANN learning algorithm can be described as a set of procedures. The procedures on the first level are responsible for solving their local tasks and calculating the part of matrix weight coefficients. The second-level procedure has to coordinate all the local procedures (tasks) using its own local target function. The third-level procedure calculates the learning parameters which are used by the second-level. There is a vertical decomposition and interaction between the procedures. Two types of information are sent between the levels. From the second level to the first level, one is a downward transmission of control signals:

$$\Gamma = (\Gamma^1, \Gamma^2, ... \Gamma^{L-1}) \tag{16}$$

Where:
Γ^l - vector of data sent from the coordinator to the two neighboring sub-networks l and $l+1$,
$l = 1 \div (L-1)$ - number of sub-networks,

From the first level to the second level two sets of feedback signals are sent:

- forward feedback errors, which are generated by every sub-network when all sub-networks calculate their local target functions value Φ^l:

$$\mathcal{E}_F = (\mathcal{E}_F^1, \mathcal{E}_F^2, ... \mathcal{E}_F^{L-1}) \tag{17}$$

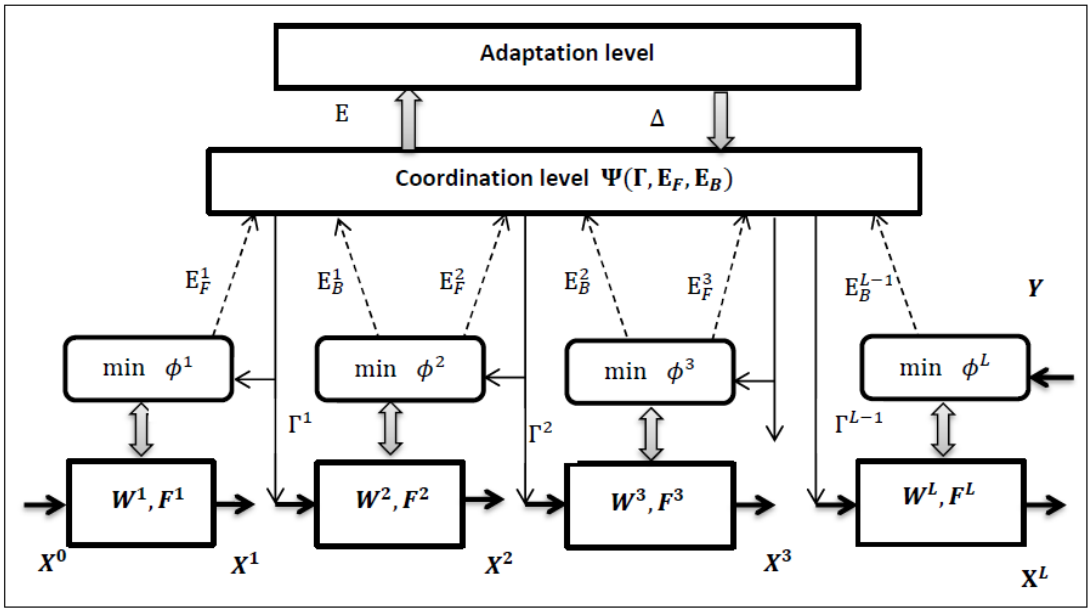


Figure 3: A scheme of an error back propagation through the layers

- backward feedback errors, which are calculated by every sub-network in the back propagation procedure:

$$\mathcal{E}_B = (\mathcal{E}_B^1, \mathcal{E}_B^2, ...\mathcal{E}_B^L) \tag{18}$$

2.1 Levels of calculation complexity

The standard ANN learning algorithm is a non-linear minimization task without constraints. To solve this task, iteration procedures are used. Using the most popular back propagation algorithm, one has to choose a lot of control parameters. The algorithm is time-consuming and its convergence is not fast. Dividing the primary algorithm into the sub-network tasks, the local target functions are simpler and can be used in different procedures. Additionally, a new procedure is needed: the coordination procedure. In practice, however, the coordinator does not have the ability to find all the parameters needed for the first-level procedures. To solve this problem, a multi-level decision hierarchy is proposed [1]. The problems is solved by the iteration algorithm on both the first and the second level. One can observe specific dynamic processes. These processes are non-linear and use a lot of control parameters. During the learning process these parameters are stable and do not change. Practice proves that this solution is not optimal. To control the way learning parameters are changed in the iteration process, an additional level could be used - the adaptation level (Fig. 3). Thus, one can build three levels as a minimum:

- The local optimization procedures: the algorithm is defined directly as a minimization task without constraints.
- The coordination procedure: this algorithm could be defined directly as a minimization of the target function as well. Constraints could exist or not.

- The adaptation procedure: the task or procedure on this level should specify the value of learning parameters not only for the coordinator level, but also on the first level. To solve this task, a procedure should achieve dynamic characteristic of the learning process from all the levels.

As a conclusion, one can state that the complexity of the problem increases from the first level to the next one. The coordination and adaptation procedures need more time to solve their own procedure.

3 Calculation algorithm structure

The deep neural network with an input layer, a set of hidden layers and an output layer can be used for further considerations. As standard $ReLu$ activation function is used for all the hidden layers. For an output layer both sigmoid or $ReLu$ activation functions could be used. A three part learning algorithm will be considered to decompose the standard learning algorithm's structure into sub-network tasks and taking into account Fig. 3.

3.1 Forward calculation

All sub - networks are independent because the coordinator sent an input and an output vector Γ^l for all the layers. Sub-networks can calculate all values in a parallel way:

- For the first sub-network:

$$\Phi^1(W^1, X^0, \Gamma^1) = \frac{1}{2} \cdot (X^1 - \Gamma^1)^T \cdot (X^1 - \Gamma^1) \tag{19}$$

Others relations:

$$X^1 = F(U^1) \tag{20}$$

$$U^1 = W^1 \cdot X^0 \tag{21}$$

Where: X^0, X^1 - input and output vector of the first layer, accordingly,
F - vector of activation function. $ReLu = max(0, U^1)$,
U^1 - internal vector.

- For the all hidden layers $l = 2 : (L - 1)$:

$$\Phi^l(W^1, \Gamma^{l-1}, \Gamma^l) = \frac{1}{2} \cdot (X^l - \Gamma^l)^T \cdot (X^l - \Gamma^l) \tag{22}$$

$$X^l = F(U^l) \tag{23}$$

$$U^l = W^l \cdot \Gamma^{l-1} \tag{24}$$

Where:
X^{l-1}, X^l - input and output vector of all the hidden layers, accordingly,
F - vector of activation function. $ReLu = max(0, U^l)$,
U^1 - internal vector.

- For the output layer:

$$\Phi^L(W^L, \Gamma^{L-1}, Y) = \frac{1}{2} \cdot (X^L - Y)^T \cdot (X^L - Y) \tag{25}$$

$$X^L = F(U^L) \tag{26}$$

$$U^L = W^L \cdot \Gamma^{L-1} \tag{27}$$

Where:
X^L - output vector,
Y - learning vector P epoch included.

3.2 Backward calculation

When all sub-networks have finished the forward calculation process, the next step can begin, that is the backward calculation. The calculated error will be sent to the coordinator. Modified formulas from subsection 1.1 are used. All subnetworks with their own target functions can calculate their own backward errors in a parallel way:

- For the first sub-network, only the forward error is calculated,(see Fig.4):

$$\mathcal{E}_F^1 = X^1 \tag{28}$$

From formula(19) partial derivatives for the matrix of weight coefficient W^1 are:

$$\frac{\partial \Phi^1}{\partial W^1} = \frac{\partial \Phi^1}{\partial X^1} \cdot \frac{\partial X^1}{\partial U^1} \cdot \frac{\partial U^1}{\partial W^1} \tag{29}$$

$$\frac{\partial \Phi^1}{\partial X^1} = E^1 = X^1 - \Gamma^1 \tag{30}$$

Formula (29) can be rewritten in the matrix form:

$$\frac{\partial \Phi^1}{\partial W^1} = \{E^1 \odot 1(U^1)\} \cdot (X^0)^T \tag{31}$$

Where:
$1(U^1)$ - derivative of the $ReLu$ activation function $ReLu^{'} = max(0, U^1)^{'} = 1(U^1)$.
The first sub-network can calculate a new value of the matrix weight coefficients W^1:

$$W^1(n+1) = W^1(n) - \alpha \cdot \frac{\partial \Phi^1}{\partial W^1} \tag{32}$$

Where:
n - current iteration number.

- For all the hidden layer $l = 2 : (L-1)$ and using formula (22), partial derivatives for the matrix of weight coefficient W^l and an input vector from the coordinator Γ^{l-1} are:

$$\frac{\partial \Phi^l}{\partial \Gamma^{l-1}} = \frac{\partial \Phi^l}{\partial X^l} \cdot \frac{\partial X^l}{\partial U^l} \cdot \frac{\partial U^l}{\partial \Gamma^{l-1}} \tag{33}$$

$$\frac{\partial \Phi^l}{\partial X^l} = E^l = X^l - \Gamma^l \tag{34}$$

$$\frac{\partial U^l}{\partial \Gamma^{l-1}} = W^l \tag{35}$$

Formula (33) can be rewritten in the matrix form:

$$\frac{\partial \Phi^l}{\partial \Gamma^{l-1}} = \mathcal{E}^{l-1} = (W^l)^T \cdot \{1(U)^l \odot \mathcal{E}^l\} \tag{36}$$

Where:
$1(U^l)$ - derivative of the $ReLu$ activation function $ReLu^{'} = max(0, U^l)^{'} = 1(U^l)$,

$$\mathcal{E}_B^{l-1} = \Gamma^l - \beta \cdot \frac{\partial \Phi^l}{\partial \Gamma^{l-1}} = \Gamma^l - \beta \cdot \mathcal{E}^{l-1} \tag{37}$$

The derivative for the weight coefficients of matrix W^l is:

$$\frac{\partial \Phi^l}{\partial W^l} = \frac{\partial \Phi^l}{\partial X^l} \cdot \frac{\partial X^l}{\partial U^l} \cdot \frac{\partial U^l}{\partial W^l} \tag{38}$$

$$\frac{\partial U^l}{\partial W^l} = \Gamma^{l-1} \tag{39}$$

Formula (38) can be rewritten i the matrix form:

$$\frac{\partial \Phi^l}{\partial W^l} = \{E^l \odot 1(U^l)\} \cdot (\Gamma^{l-1})^T \tag{40}$$

A sub-network can calculate the new value of the matrix weight coefficients:

$$W^l(n+1) = W^l(n) - \alpha \cdot \frac{\partial \Phi^l}{\partial W^l} \tag{41}$$

Where:
n - current iteration number.

- For the output layer using formula (25) partial derivatives for the matrix of weight coefficient W^L and an input vector from the coordinator Γ^{l-1} are:

$$\frac{\partial \Phi^L}{\partial \Gamma^{L-1}} = \frac{\partial \Phi^L}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial \Gamma^{L-1}} \tag{42}$$

$$\frac{\partial \Phi^L}{\partial X^L} = \mathcal{E}^L = X^L - Y \tag{43}$$

$$\frac{\partial U^L}{\partial \Gamma^{L-1}} = W^L \tag{44}$$

Formula (42) can be rewritten in the matrix form:

$$\frac{\partial \Phi^L}{\partial \Gamma^{L-1}} = \mathcal{E}^{L-1} = (W^L)^T \cdot \{1(U)^L \odot \mathcal{E}^L\} \tag{45}$$

$$\mathcal{E}_B^{l-1} = \Gamma^{L-1} - \beta \cdot \frac{\partial \Phi^L}{\partial \Gamma^{L-1}} = \Gamma^{L-1} - \beta \cdot \mathcal{E}^{L-1} \tag{46}$$

Applying the same procedure as above, a set of formulas is written:

$$\frac{\partial \Phi^L}{\partial W^L} = \frac{\partial \Phi^l}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial W^L} \tag{47}$$

$$\frac{\partial U^L}{\partial W^L} = \Gamma^{L-1} \tag{48}$$

Formula (47) in the matrix form:

$$\frac{\partial \Phi^L}{\partial W^L} = \{\mathcal{E}^L \odot 1(U^L)\} \cdot (\Gamma^{L-1})^T \tag{49}$$

The output sub-network can calculate the new value of the matrix weight coefficients:

$$W^L(n+1) = W^L(n) - \alpha \cdot \frac{\partial \Phi^L}{\partial W^L} \tag{50}$$

Where:
n - current iteration number.

When all sub-networks have finished calculation of their local target functions, the forward $\mathcal{E}_F^l$ and backward $\mathcal{E}_B^l$ feedback information are sent to the coordinator. At the moment all the sub- networks modified of their matrixes of weight coefficient W^l for $l = 1 : L$.

4 Coordinator structure

In a hierarchical learning algorithm, the coordinator plays the main role. It is now time to decide what kind of coordination principle will be chosen. This principle specifies various strategies for the coordinator and determines the structure of the coordinator. In [1] three methods were introduced in which the interaction could be performed:

- Interaction Prediction. The coordination input may involve a prediction of the interface input.
- Interaction Decoupling. Each first-level sub-system is introduced into the solution of its task and can treat the interface input as an additional decision variable to be free. Consequently, sub-systems are completely decoupled.
- Interaction Estimation. The coordinator specifies the ranges of interface inputs over which they may vary.

For a deep neural network containing L layers, the coordinator prepare $l = 1 \div (L-1)$ coordination signal Γ^l. One of them is treated as input vectors and the other as learning data vectors for local target functions Φ^l for $l = 1 : (L-1)$. Every coordinator signal is correlated with two feedback signals $\mathcal{E}_F^l$ and $\mathcal{E}_B^l$. These signals are calculated by the first level sub-networks and are sent to coordinator. The coordinator uses its own target function:

$$\Psi = \frac{1}{2} \sum_{i=1}^{i=L-1} \{(\Gamma^i - \mathcal{E}_B^i)^T \cdot (\Gamma^i - \mathcal{E}_B^i)\} + \{(\Gamma^i - \mathcal{E}_F^i)^T \cdot (\Gamma^i - \mathcal{E}_F^i)\} \tag{51}$$

To minimize this function, the gradient method is used:

$$\frac{\partial \Psi}{\partial \Gamma^i} = (\Gamma^i - \mathcal{E}_B^i) + (\Gamma^i - \mathcal{E}_F^i) = 2 \cdot \Gamma^i - [\mathcal{E}_B^i + \mathcal{E}_F^i] \tag{52}$$

The new value of coordination signal is calculated using the gradient method:

$$\Gamma^i(n+1) = \Gamma^i(n) - \rho \cdot \frac{\partial \Psi}{\partial \Gamma^i} \tag{53}$$

This new coordinator signals value is sent to the first level sub-networks and the entire process is repeated.

5 Adaptation level

In formula (53) the learning parameter ρ could be constant or could change during the iteration process. In the beginning the learning process is very dynamic, a large oscillation could be seen in the first level of the local target functions. Because parameter ρ has to be small, the iteration process is stable. However, although the minimum Ψ is achieved asymptotically, it happens very slowly. To improve this algorithm, the coordinator sends the target function value $\Psi(n)$ to the adaptation level. The following simple algorithm could be used:
If $\Psi(n) > \Psi(n-1)$ then $\rho = k_d \cdot \rho$, where: $k_d = 0.95 \div 0.98$ - decreasing parameter,
If $\Psi(n) < \Psi(n-1)$ then $\rho = k_i \cdot \rho$, where: $k_i = 1.02 \div 1.05$ - increasing parameter. The new ρ is sent to the coordinator and used in the iteration process. To make the entire learning process more stable, Ψ is usually averaged by epoch:

$$\Psi(n, p+1) = \frac{1}{N_p} \cdot \Psi(n, p) + \frac{N_p - 1}{N_p} \cdot \sum_{p=1}^{p=N_p-1} \Psi(n, i) \tag{54}$$

Where: N_p - number of input vector in epoch, p - actual index value.

6 Numerical example and conclusion

In a life insurance company the underwriting process has been playing the central role in risk control and premium calculation. A deep neural network could be used to help the insurance agents to classify the insurance applicant and calculate the first version of premium. Therefore, a special short questionnaire was prepared which includes only 10 main questions (Fig.4). All the data were divided into three subsets: - learning set includes 250 records, -verification set includes 50 vectors,- testing includes 100 vectors.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
Nr	Sex	Years old	Weight in kg	Growth in cm	Smoking	Father's long life	Mather's long life	Children number	Living condition	Living place
1	1	48	70	190	0	68	67	0	1	2
2	1	62	71	182	0	82	91	1	0	0
3	0	64	83	180	5	89	100	2	1	2
4	1	40	107	177	10	74	66	2	1	2
5	1	58	120	183	20	90	77	3	1	1

Figure 4: An input data structure example

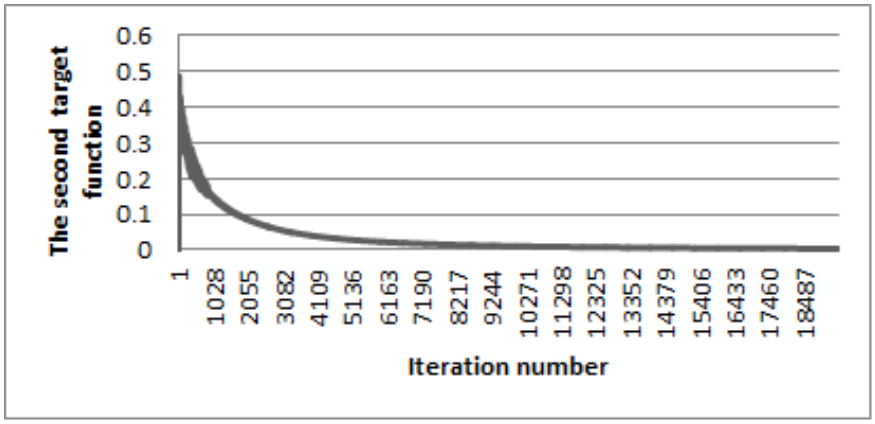


Figure 5: Dynamic characteristic of the second target function Φ^2

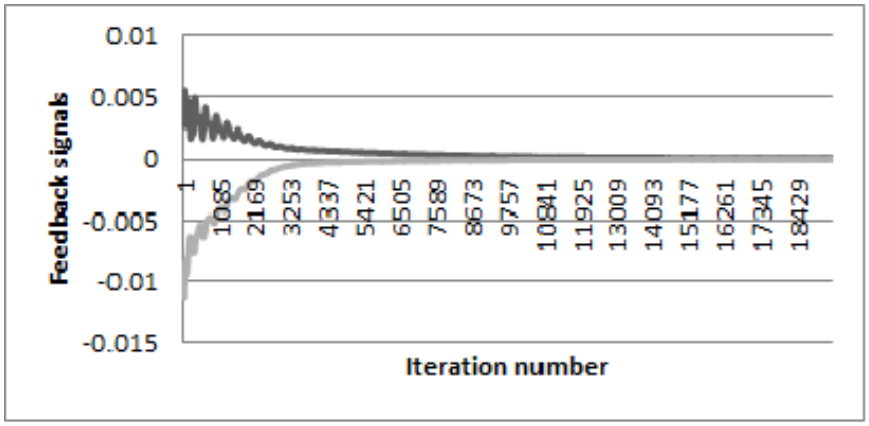


Figure 6: Dynamic characteristics of the two feedback signals $\mathcal{E}_B^2$ and $\mathcal{E}_F^2$

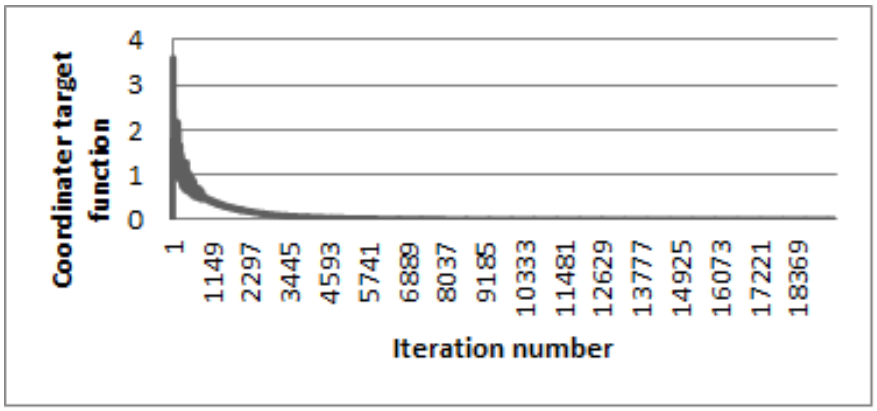


Figure 7: Dynamic characteristics of the coordinator

Dynamic characteristics of the second subnetwork contain two phases. From start to 4000 iteration, error decrease very fast. In the next phase, the learning process is not optimal.

In Fig. 6. the feedback signals do not have optimal characteristics. Probably the learning coefficient ρ is too hight and the coordinator tries to accelerate the learning process. Result is reverse. The adaptation level should force its own strategy to stabilize the learning process. Future works should focus on the coordinator and the adaptation level strategy. The main question is - how to improve the characteristics of the learning process?

References

[1] M. D. Mesarocic, D. Macko, and Y. Takahara, Theory of hierarchical multilevel systems, Academic Press, New York and London, 1970.

[2] Ch. M. Bishop, Pattern Recognition and Machine Learning, Springer Science + Business Media, LLC 2006.

[3] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016

[4] Zeng-Guang Hou, Madan M. Gupta, Peter N. Nikiforuk, Min Tan, and Long Cheng, "A Recurrent Neural Network for Hierarchical Control of Interconnected Dynamic Systems", IEEE Transactions on Neural Networks, Vol. 18, No. 2, March 2007.

[5] Joarder Kamruzzaman, Rezaul Begg, Artificial Neural Network in Finance and Manufacturing, Idea Group Publishing, Hershley, Pennsylvania 2006.

[6] S.K. Zhou, H. Greenspan, D. Shen, Deep Learning for Medical Image Analysis, Academic Press 2017

[7] M. Nielson, Neural Network and Deep Learning, Determination Press, 2015

[8] S. Placzek, B. Adhikari, "Analysis of Multilayer Neural Network with Direct Connection Cross-forward Connection", CS&P Conference 2013, The University of Warsaw, Warsaw 2013.

[9] L. Rutkowski, Metody i techniki sztucznej inteligencji, Wydawnictwo Naukowe PWN, Warszawa 2006.

Author Index

Akhundov, Jafar 159
Artiemjew, Piotr 97

Barbuti, Roberto 119
Barczak, Andrzej 15
Barczak, Michał 15
Bove, Pasquale 119
Burzańska, Marta 57

Czaja, Ludwik 27
Czaus, Przemysław 69

Dutta, Soma 107

Franczyk, Bogdan 57

Getir, Sinem 81
Gori, Roberta 119, 133
Grabowski, Adam 39
Gribovskaya, Nataliya 145
Grunske, Lars 81
Gruska, Damas 3, 133

Levi, Francesca 119

Milazzo, Paolo 119, 133

Nguyen, Linh Anh 171

Pavese, Esteban 81
Penczek, Wojciech 103
Polkowski, Lech 47
Płaczek, Aleksander 209
Płaczek, Stanisław 209

Redziejowski, Roman 91
Reißner, Michael 159
Ropiak, Krzysztof 197
Rozenberg, Grzegorz 107
Ruiz, M. Carmen 3

Sidoruk, Teofil 73
Skowron, Andrzej 107

Virbitskaite, Irina 145

Weidlich, Matthias 1
Werner, Matthias 159

Zimniak, Marcin 57
Zmudzinski, Lukasz 185

Keyword Index

animal recognition 185
automated tests 69

Backus Naur Form 91
betweenness 47
big data 69
Bisimulation 145
Boolean satisfiability 73

cause-effect structures 27
Classification 197
classifiers 69
CNF 73
Comparative Semantics 145
composition 159
compositional expressiveness 159
compositionality 159
concurrency 27
coordination procedure 209
CSG Classifier 197

datasets 69
Decision Systems 197
decomposition and coordination 209
deep learning 185
deep neural network 209
description logics 171
DIMACS 73
Distributed Systems 15

Ensemble Model 197
Event Structures 145
exploration system 107
expressive power 171
expressiveness 159

formalization of mathematics 39
formula based predictors 133

Gene Regulatory Networks 119
Gossip Algorithm 15

heuristic methods 57
hierarchical structure 209

Homogenous Granulation 197
hybrid automata 159
hybrid systems 159

infomorphism 107
information flow 3, 133
information spreading in distributed systems 15
information system 107

left recursion 91
local logic 107
logic infomorphism 107

mass 47
Mizar Mathematical Library 39

natural semantics 91

opacity 3, 133
optimization in physical database design 57

parsing 91
Parsing Expression Grammars 91
periodic patterns 57
periodic patterns discovery 57
Petri nets 27
probabilistic formalism 81
probabilistic model checking 81
probabilistic verification 81
process algebras 3
Propositional Dynamic Logic (PDL) 171
Propositional formula 73

Random Granular Reflections 197
Reaction Systems 119, 133, 107
robotics 185
rough approximation 39
rough inclusion 47
Rough Sets 197

SAT 73
SAT-solvers 73
security 3, 133
stochastic regular expressions 81
supervisory control 3

Test-Free PDL 171
the Bayes theorem 47

Threshold Boolean networks 119
Transition Systems 145

workload 57
workload reconstruction 57

Yeast-Cell Cycle 119

zoom structure 107