Joint Proceedings of:

»LAM'12«

5th International Workshop on

Logics, Agents, and Mobility

»WooPS 2012«

1st International Workshop on

Petri Net-based Security

»CompoNet 2012«

2nd International Workshop on

Petri Nets Compositions

Satellite events of the

33th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency

and the

12th International Conference on Application of Concurrency to System

Hamburg, Germany, June, 2012

Compilation Editor:
Michael Köhler-Bußmeier
University of Hamburg
Department for Informatics
Theoretical Foundations of Informatics
Vogt-Kölln-Str. 30
D-22527 Hamburg
Germany
http://www.informatik.uni-hamburg.de/TGI/

Table of Contents

Preface by the LAM'12 Organisers

This volume contains of the contributions to the *5th International Workshop on Logics, Agents, and Mobility*. The workshop took place as a satelite event of 33th International Conference on Application and Theory of Petri Nets and Other Models of Concurrency and the 12th International Conference on Application of Concurrency to System in Hamburg, Germany.

The aim of this series of workshops is to bring together active researchers in the areas of logics and other formal frameworks on the one hand, and mobile systems on the other hand. The main focus is on the field of logics and calculi for mobile agents, and multi-agent systems. Many notions used in the theory of agents are derived from philosophy, logic, and linguistics, and interdisciplinary discourse has proved fruitful for the advance of this domain.

Outside of academia, the deployment of large-scale pervasive infrastructures (mobile ad-hoc networks, mobile devices, RFIDs, etc.) is becoming a reality. This raises a number of scientific and technological challenges for the software modelling and programming models for such large-scale, open and highly-dynamic distributed systems. The agent and multi-agent systems approach seems particularly adapted to tackle this challenge, but there are many issues remaining to be investigated. For instance, the agents must be location-aware since the actual services available to them may depend on their (physical or virtual) location. The quality and quantity of resources at their disposal is also largely fluctuant, and the agents must be able to adapt to such highly dynamic environments. Moreover, mobility itself raises a large number of difficult issues related to safety and security, which require the ability to reason about the software. Logics and type systems with temporal or other kinds of modalities (relating to location, resource and/or security-awareness) play a central role in the semantic characterisation and then verification of properties about mobile agent systems.

There are still many open problems and research questions in the theory of such systems. The workshop is intended to showcase results and current work being undertaken in these areas with a focus on logics for specification and verification of dynamic, mobile systems.

We would like to thank all authors who have submitted papers. Each paper was reviewed by at least three referees. During the reviewing process the program comittee selected three contributions for publication. We wish to thank all members of the program comittee for their effort.

Finally we would like to thank our invited speaker, Julia Padberg, for her lecture: *Reconfigurable Petri Nets: Modeling and Analysis.*

June 2012

Berndt Müller (Farwer)
Michael Köhler-Bußmeier

Programme Chairs

Berndt Müller (Farwer)
Michael Köhler-Bußmeier

Programme Committee

Matteo Baldoni
Nils Bulling
Marina De Vos
Louise Dennis
Jürgen Dix
Michael Fisher
Didier Galmiche
Andreas Herzig
Kathrin Hoffmann
Michael Köhler-Bußmeier
Joao Leite
Dale Miller
Berndt Müller (chair)
Frederic Peschanski
Wamberto Vasconcelos
Thomas Ågotnes

Reconfigurable Petri Nets: Modeling and Analysis

Julia Padberg

Hamburg University of Applied Sciences, Germany

Abstract
The results of the research group forMA$_\ell$NET (funded by the German Research Council 2006-2012) on reconfigurable Petri nets are presented in this talk. We introduce a family of modeling techniques consisting of Petri nets together with a set of rules. For reconfigurable Petri nets not only the follower marking can be computed but also the structure can be changed by rule application to obtain a new net that is more appropriate with respect to some requirements of the environment. Moreover, these activities can be interleaved. For rule-based modification of Petri nets we use the framework of net transformations that is inspired by graph transformation systems. The basic idea behind net transformation is the stepwise modification of Petri nets by given rules. The rules present a rewriting of nets where the left-hand side is replaced by the right-hand side.

Motivation for this family of formal modeling techniques is the observation that in increasingly many application areas the underlying system has to be dynamic in a structural sense. Complex coordination and structural adaptation at run-time (e.g. mobile ad-hoc networks, communication spaces, ubiquitous computing) are main features that need to be modeled adequately. The distinction between the net behavior and the dynamic change of its net structure is the characteristic feature that makes reconfigurable Petri nets so suitable for systems with dynamic structures.

In this talk we first motivate the use of reconfigurable Petri nets and present their basic ideas. The concepts are discussed and are then given mathematically. We employ the notion of high-level replacement systems extensively to obtain rules and transformations of place/transition nets. These notions are then exemplified in a case study modeling scenarios of the *Living Place Hamburg*. Living Place Hamburg can be considered as a system of ubiquitous computing and ambient intelligence. Scenarios of the Living Place are modeled using reconfigurable place/transition nets providing a formal model of the internal system behavior of this system, so that this model helps us to improve our understanding of the modeled system. Subsequently we extend the theory to algebraic high-level nets, this employs again high-level replacement systems with nested application conditions as well as a individual token approach. These new concepts are illustrated and discussed in the case study *Skype* at length. The analysis of *Skype* as a concrete and typical existing Communication Platform is an example for a modeling methodology for Communication Platforms using an integrated modeling approach based on algebraic higher order nets.

The talk is concluded with a discussion of further results.

Modelling Intentional Reasoning with Defeasible and Temporal Logic

José Martín Castro-Manzano

Escuela de Filosofía
Universidad Popular Autónoma del Estado de Puebla
21 sur 1103 Barrio de Santiago, Puebla, México 72410
Instituto de Investigaciones Filosóficas
Universidad Nacional Autónoma de México
Circuito Mario de la Cueva s/n Ciudad Universitaria, México, D.F., México 04510
josemartin.castro@upaep.mx

Abstract. We follow the hypothesis that intentional reasoning is a form of logical reasoning *sui generis* by its double nature: temporal and defeasible. Then we briefly describe a formal framework that deals with these topics and we study the metalogical properties of its notion of inference. The idea is that intentional reasoning can be represented in a well-behaved defeasible logic and has the right to be called logical reasoning since it behaves, *mutatis mutandis*, as a logic, strictly speaking, as a non-monotonic logic.

Keywords: Defeasible logic, temporal logic, BDI logic, intention.

1 Introduction

The relationship between philosophy and computer science is very profound and unique [23]. Not only because these disciplines share some common historical data –like Leibniz's *mathesis universalis* [8]– and interesting anecdotes –like the correspondence between Newell and Russell [11]–, but more importantly because from the constant dialog that occurs within these disciplines we gain useful hypothesis, formal methods and functional analysis that may shed some light about different aspects of the nature of human behavior, specially under a cognitive schema. The cognitive schema we follow is the BDI model (that stands for *Beliefs*, *Desires* and *Intentions*) as originally exposed by Bratman [4] and formally developed by Rao and company [21,22]. The general aspect we study is the case of the non-monotonicity of intentional reasoning.

There is no doubt that reasoning using beliefs and intentions during time is a very common task, done on a daily basis; but the nature and the status of such kind of reasoning, which we will be calling intentional, are far from being clear and distinct. However, it would be blatantly false to declare that this study is entirely new, for there are recent efforts to capture some of these ideas already [13,16,19]. But, in particular, we can observe, on one side, the case of BDI logics [22,24] in order to capture and understand the nature of intentional

reasoning; and on the other side, the case of defeasible logics [20] to try to catch the status of non-monotonic reasoning.

The problem with these approaches, nevertheless, is that, in first place, human reasoning is not and should not be monotonic [18], and thus, the logical models should be non-monotonic, but the BDI techniques are monotonic; and in second place, intentional states should respect temporal norms, and so, the logical models need to be temporal as well, but the non-monotonic procedures do not consider the temporal or intentional aspect. So, in the state of the art, defeasible logics have been mainly developed to reason about beliefs [20] but have been barely used to reason about temporal structures [14]; on the other hand, intentional logics have been mostly used to reason about intentional states and temporal behavior but most of them are monotonic [7,21,24].

Under this situation our main contribution is a brief study of the nature and status of intentional reasoning following the hypothesis that intentional reasoning is a form of logical reasoning *sui generis* by its temporal and defeasible nature and we suggest that intentional reasoning has the right to be called *logical* since it behaves, *mutatis mutandis*, as a logic. In particular, this study is important by its own sake because defeasible reasoning has certain patterns of inference and therefore the usual challenge is to provide a reasonable description of these patterns. Briefly, the idea is that if monotony is not a property of intentional reasoning and we want to give an adequate description of its notion of inference, then we must study the metalogical properties of intentional inference that occur instead of monotony. Because once monotonicity is given up, a very organic question about the status of this kind of reasoning emerges: why should we consider intentional reasoning as an instance of a logic *bona fide*?

This paper is organized in the next way. In Section 2 we briefly expose what is understood as intentional reasoning. In Section 3 is our main contribution and finally, in Section 4 we sum up the results obtained.

2 Intentional reasoning

Two general requirements to be checked out while developing a logical framework are material and formal adequacy [1]. Material adequacy is about capturing an objective phenomenon. Formal adequacy has to do with the metalogical properties that a notion of logical consequence satisfies. The nature of intentional reasoning is related to a material aspect, while its status is directly connected with a formal one. During this study, due to reasons of space, we will focus mainly on the latter in order to argue that intentional reasoning can be modelled in a well-behaved defeasible logic, given that a well-behaved defeasible logic has to satisfy conditions of Supraclassicality, Reflexivity, Cut and Cautious Monotony [12].

But just to give some pointers about material adequacy, let us consider the next example for sake of explanation: assume there is an agent that has an intention of the form $on(X, Y) \leftarrow put(X, Y)$. This means that, for such an agent to achieve $on(a, b)$ it typically has to put a on b. If we imagine such an agent is immersed in a dynamic environment, of course the agent will try to put, typ-

ically, a on b; nevertheless, a *rational* agent would only do it as long as it is *possible*; otherwise, we would say the agent is not rational. Therefore, it results quite natural to talk about some intentions that are maintained typically but not absolutely if we want to guarantee some level of rationality. And so, it is reasonable to conclude that intentions –in particular policy-based intentions [4]–, allow some form of defeasible reasoning [13] that must comply with some metalogical properties. But before we explore such properties, let us review some previous details.

The current logical systems that are used to model intentional reasoning are built in terms of what we call a bratmanian model. A bratmanian model is a model that *i)* follows general guidelines of Bratman's theory of practical reasoning [4], *ii)* uses the BDI architecture [21] to represent data structures and *iii)* configures notions of logical consequence based on relations between intentional states. There are several logics based upon bratmanian models, but we consider there are, at least, two important problems with the usual logics [7,22,24].

For one, such logics tend to interpret intentions as a unique fragment –usually represented by an opertator INT–, while Bratman's original theory distinguished three classes of intentions: deliberative, non-deliberative and policy-based. In particular, policy-based intentions are of great importance given their structure and behavior: they have the structure of complex rules and behave like plans. This remark is important for two reasons: because the existing formalisms, despite of recognizing the intimate relationship between plans and intentions, seem to forget that intentions behave like plans; and because the rule-like structure allows us to build a more detailed picture of the nature of intentional reasoning.

But the bigger problem is that these systems do not quite recognize that intentional reasoning has a temporal and defeasible nature. Intuitively, the idea is that intentional reasoning is temporal because intentions and beliefs are dynamic data structures, i.e., they change during time; but is also defeasible, because if these data structures are dynamic, their consequences may change. The bratmanian model we propose tries to respect this double nature by following the general guidelines of Bratman's theory of practical reasoning [4], so we distinguish functional (pro-activity, inertia, admissibility), descriptive (partiality, dynamism, hierarchy) and normative (internal, external consistency and coherence) properties that configure a notion of inference. To capture this notion of inference in a formal fashion the next framework is proposed in terms of $AgentSpeak(L)$[3] (see Appendix):

Definition 1 *(Non-monotonic intentional framework) A non-monotonic intentional framework is a tuple* $\langle B, I, F_B, F_I, \vdash, \mid\!\sim, \dashv, \sim\!\mid, \succ\rangle$ *where:*

- B *denotes the belief base.*
- I *denotes the set of intentions.*
- $F_B \subseteq B$ *denotes the basic beliefs.*
- $F_I \subseteq I$ *denotes the basic intentions.*
- $\vdash$ *and* $\dashv$ *are strong consequence relations.*
- $\mid\!\sim$ *and* $\sim\!\mid$ *are weak consequence relations.*

– $\succ \subseteq I^2$ *s.t.* $\succ$ *is acyclic.*

The item B denotes the beliefs, which are literals. F_B stands for the beliefs that are considered as basic; and similarly F_I stands for intentions considered as basic. Each intention $\phi \in I$ is a structure $te : ctx \leftarrow body$ where te represents the goal of the intention –so we preserve *proactivity*–, ctx a context and the rest denotes the body. When ctx or $body$ are empty we write $te : \top \leftarrow \top$ or just te. Also it is assumed that plans are *partially* instantiated.

Internal consistency is preserved by allowing the context of an intention denoted by $ctx(\phi)$, $ctx(\phi) \in B$ and by letting te be the head of the intention. So, *strong consistency* is implied by internal consistency (given that strong consistency is $ctx(\phi) \in B$). *Means-end coherence* will be implied by *admissibility* –the constraint that an agent will not consider contradictory options– and the *hierarchy* of intentions is represented by the order relation, which we require to be acyclic in order to solve conflicts between intentions. And with this framework we can arrange a notion of inference where we will say that ϕ is strongly (weakly) derivable from a sequence Δ if and only if there is a proof of $\Delta \vdash \phi$ ($\Delta \mid\sim \phi$). And also, that ϕ is not strongly (weakly) provable if and only if there is a proof of $\Delta \dashv \phi$ ($\Delta \sim\!\!\mid \phi$), where $\Delta = \langle B, I \rangle$.

2.1 The system $NBDI^{CTL}_{AS(L)}$

We start with $CTL_{AgentSpeak(L)}$ [15] as a logical tool for the formal specification (similar approaches have been accomplished for other programming languages [9]). Of course, initially, the approach is similar to a BDI^{CTL} system defined after $B^{KD45}D^{KD}I^{KD}$ with the temporal operators: *next* ($\bigcirc$), *eventually* ($\Diamond$), *always* ($\Box$), *until* (U), *optional* (E), *inevitable* (A), and so on, defined after $CTL*$ [6,10].

Syntax of $BDI^{CTL}_{AS(L)}$ $CTL_{AgentSpeak(L)}$ may be seen as an instance of BDI^{CTL}. The idea is to define some BDI^{CTL} semantics in terms of $AgentSpeak(L)$ structures. So, we need a language able to express temporal and intentional states. Thus, we require in first place some way to express these features.

Definition 2 *(Syntax of $BDI^{CTL}_{AS(L)}$) If ϕ is an $AgentSpeak(L)$ atomic formula, then $\mathsf{BEL}(\phi)$, $\mathsf{DES}(\phi)$ and $\mathsf{INT}(\phi)$ are well formed formulas of $BDI^{CTL}_{AS(L)}$.*

To specify the temporal behavior we use $CTL*$ in the next way.

Definition 3 *($BDI^{CTL}_{AS(L)}$ temporal syntax) Every $BDI^{CTL}_{AS(L)}$ formula is a state formula s:*

– $s ::= \phi | s \wedge s | \neg s$
– $p ::= s | \neg p | p \wedge p | \mathsf{E}p | \mathsf{A}p | \bigcirc p | \Diamond p | \Box p | p \ \mathsf{U} \ p$

Semantics of $BDI^{CTL}_{AS(L)}$ Initially the semantics of BEL, DES and INT is adopted from [2]. So, we assume the next function:

$$\begin{aligned} agoals(\top) &= \{\}, \\ agoals(i[p]) &= \begin{cases} \{at\} \cup agoals(i) & \text{if } p = +!at : ct \leftarrow h, \\ agoals(i) & \text{otherwise} \end{cases} \end{aligned}$$

which gives us the set of atomic formulas (at) attached to an achievement goal ($+!$) and $i[p]$ denotes the stack of intentions with p at the top.

Definition 4 *($BDI^{CTL}_{AS(L)}$ semantics) The operators* BEL, DES *and* INT *are defined in terms of an agent ag and its configuration $\langle ag, C, M, T, s\rangle$:*

$$\mathsf{BEL}_{\langle ag,C,M,T,s\rangle}(\phi) \equiv \phi \in bs$$

$$\mathsf{INT}_{\langle ag,C,M,T,s\rangle}(\phi) \equiv \phi \in \bigcup_{i \in C_I} agoals(i) \vee \bigcup_{\langle te,i\rangle \in C_E} agoals(i)$$

$$\mathsf{DES}_{\langle ag,C,M,T,s\rangle}(\phi) \equiv \langle +!\phi, i\rangle \in C_E \vee \mathsf{INT}(\phi)$$

where C_I denotes current intentions and C_E suspended intentions.

And now some notation: we will denote an intention ϕ with head g by $\phi[g]$. Also, a negative intention is denoted by $\phi[g^c]$, i.e., the intention ϕ with $\neg g$ as the head. The semantics of this theory will require a Kripke structure $K = \langle S, R, V\rangle$ where S is the set of agent configurations, R is an access relation defined after the transition system Γ and V is a valuation function that goes from agent configurations to true propositions in those states.

Definition 5 *Let $K = \langle S, \Gamma, V\rangle$, then:*

- *S is a set of agent configurations $c = \langle ag, C, M, T, s\rangle$.*
- *$\Gamma \subseteq S^2$ is a total relation such that for all $c \in \Gamma$ there is a $c' \in \Gamma$ s.t. $(c, c') \in \Gamma$.*
- *V is valuation s.t.:*
  - *$V_{\mathsf{BEL}}(c, \phi) = \mathsf{BEL}_c(\phi)$ where $c = \langle ag, C, M, T, s\rangle$.*
  - *$V_{\mathsf{DES}}(c, \phi) = \mathsf{DES}_c(\phi)$ where $c = \langle ag, C, M, T, s\rangle$.*
  - *$V_{\mathsf{INT}}(c, \phi) = \mathsf{INT}_c(\phi)$ where $c = \langle ag, C, M, T, s\rangle$.*
- *Paths are sequences of configurations $c_0, \ldots, c_n$ s.t. $\forall i (c_i, c_{i+1}) \in R$. We use x^i to indicate the i-th state of path x. Then:*

S1 $K, c \models \mathsf{BEL}(\phi) \Leftrightarrow \phi \in V_{\mathsf{BEL}}(c)$
S2 $K, c \models \mathsf{DES}(\phi) \Leftrightarrow \phi \in V_{\mathsf{DES}}(c)$
S3 $K, c \models \mathsf{INT}(\phi) \Leftrightarrow \phi \in V_{\mathsf{INT}}(c)$
S4 $K, c \models \mathsf{E}\phi \Leftrightarrow \exists x = c_1, \ldots \in K | K, x \models \phi$
S5 $K, c \models \mathsf{A}\phi \Leftrightarrow \forall x = c_1, \ldots \in K | K, x \models \phi$
P1 $K, c \models \phi \Leftrightarrow K, x^0 \models \phi$ *where ϕ is a state formula.*
P2 $K, c \models \bigcirc\phi \Leftrightarrow K, x^1 \models \phi$.
P3 $K, c \models \Diamond\phi \Leftrightarrow K, x^n \models \phi$ *for $n \geq 0$*
P4 $K, c \models \Box\phi \Leftrightarrow K, x^n \models \phi$ *for all n*
P5 $K, c \models \phi \,\mathsf{U}\, \psi \Leftrightarrow \exists k \geq 0$ *s.t.* $K, x^k \models \psi$ *and for all j, k, $0 \leq j < k | K, c^j \models \phi$ or $\forall j \geq 0 : K, x^j \models \phi$*

A notion of inference comes in four cases: if the sequence is $\Delta \vdash \phi$, we say ϕ is strongly provable; if it is $\Delta \dashv \phi$ we say ϕ is not strongly provable. If is $\Delta \mathrel{|\!\sim} \phi$ we say ϕ is weakly provable and if it is $\Delta \sim\!\!\!| \ \phi$, then ϕ is not weakly provable.

Definition 6 *(Proof) A proof of ϕ from Δ is a finite sequence of beliefs and intentions satisfying:*

1. $\Delta \vdash \phi$ *iff*
   1.1. $\Box\mathsf{A}(\mathsf{INT}(\phi))$ *or*
   1.2. $\Box\mathsf{A}(\exists\phi[g] \in F_I : \mathsf{BEL}(ctx(\phi)) \wedge \forall\psi[g'] \in body(\phi) \vdash \psi[g'])$
2. $\Delta \mathrel{|\!\sim} \phi$ *iff*
   2.1. $\Delta \vdash \phi$ *or*
   2.2. $\Delta \dashv \neg\phi$ *and*
      2.2.1. $\Diamond\mathsf{E}(\mathsf{INT}(\phi)\ \mathsf{U}\ \neg\mathsf{BEL}(ctx(\phi)))$ *or*
      2.2.2. $\Diamond\mathsf{E}(\exists\phi[g] \in I : \mathsf{BEL}(ctx(\phi)) \wedge \forall\psi[g'] \in body(\phi) \mathrel{|\!\sim} \psi[g'])$ *and*
         2.2.2.1. $\forall\gamma[g^c] \in I$, $\gamma[g^c]$ *fails at* Δ *or*
         2.2.2.2. $\psi[g'] \succ \gamma[g^c]$
3. $\Delta \dashv \phi$ *iff*
   3.1. $\Diamond\mathsf{E}(\mathsf{INT}(\neg\phi))$ *and*
   3.2. $\Diamond\mathsf{E}(\forall\phi[g] \in F_I : \neg\mathsf{BEL}(ctx(\phi)) \vee \exists\psi[g'] \in body(\phi) \dashv \psi[g'])$
4. $\Delta \sim\!\!\!| \ \phi$ *iff*
   4.1. $\Delta \dashv \phi$ *and*
   4.2. $\Delta \vdash \neg\phi$ *or*
      4.2.1. $\Box\mathsf{A}\neg(\mathsf{INT}(\phi)\ \mathsf{U}\ \neg\mathsf{BEL}(ctx(\phi)))$ *and*
      4.2.2. $\Box\mathsf{A}(\forall\phi[g] \in I : \neg\mathsf{BEL}(ctx(\phi)) \vee \exists\psi[g'] \in body(\phi) \sim\!\!\!| \ \psi[g'])$ *or*
         4.2.2.1. $\exists\gamma[g^c] \in I$ *s.t.* $\gamma[g^c]$ *succeds at* Δ *and*
         4.2.2.2. $\psi[g'] \not\succ \gamma[g^c]$

3 Formal adequacy

Once monotonicity is given up a very intuitive question arises: why should we consider intentional reasoning as an instance of a logic *bona fide*? We indirectly answer this question by arguing that intentional reasoning under this bratmanian model has some good properties.

3.1 Consistency

We suggest a square of opposition in order to depict logical relationships of consistency and coherence.

Proposition 1 *(Subalterns$_1$) If* $\vdash \phi$ *then* $\mathrel{|\!\sim} \phi$.

Proof. Let us assume that $\vdash \phi$ but not $\mathrel{|\!\sim} \phi$, i.e., $\sim\!\!\!| \ \phi$. Then, given $\vdash \phi$ we have two general cases. Case 1: given the initial assumption that $\vdash \phi$, by Definition 6 item 1.1, we have that $\Box\mathsf{A}(\mathsf{INT}(\phi))$. Now, given the second assumption, i.e., that $\sim\!\!\!| \ \phi$, by Definition 6 item 4.1, we have $\dashv \phi$. And so, $\Diamond\mathsf{E}(\mathsf{INT}(\neg\phi))$, and thus, by

the temporal semantics, we get $\neg\phi$; however, given the initial assumption, we also obtain ϕ, which is a contradiction.

Case 2: given the assumption that $\vdash \phi$, by Definition 6 item 1.2, we have that $\exists\phi[g] \in F_I : \mathsf{BEL}(ctx(\phi)) \wedge \forall\psi[g'] \in body(\phi) \vdash \psi[g']$. Now, given the second assumption, that $\sim\!\!\dashv \phi$, we also have $\dashv \phi$ and so we obtain $\Diamond\mathsf{E}(\forall\phi[g] \in F_I : \neg\mathsf{BEL}(ctx(\phi)) \vee \exists\psi[g'] \in body(\phi) \dashv \psi)$, and thus we can obtain $\forall\phi[g] \in F_I : \neg\mathsf{BEL}(ctx(\phi)) \vee \exists\psi[g'] \in body(\phi) \dashv \psi)$ which is $\neg(\exists\phi[g] \in F_I : \mathsf{BEL}(ctx(\phi)) \wedge \forall\psi[g'] \in body(\phi) \vdash \psi[g'])$. ■

Corollary 1 *(Subalterns$_2$) If $\sim\!\!\dashv \phi$ then $\dashv \phi$.*

Proposition 2 *(Contradictories$_1$) There is no ϕ s.t. $\vdash \phi$ and $\dashv \phi$.*

Proof. Assume that there is a ϕ s.t. $\vdash \phi$ and $\dashv \phi$. If $\dashv \phi$ then, by Definition 6 item 3.1, $\Diamond\mathsf{E}(\mathsf{INT}(\neg\phi))$. Thus, by proper semantics, we can obtain $\neg\phi$. However, given that $\vdash \phi$ it also follows that ϕ, which is a contradiction. ■

Corollary 2 *(Contradictories$_2$) There is no ϕ s.t. $\mid\!\sim \phi$ and $\sim\!\!\dashv \phi$.*

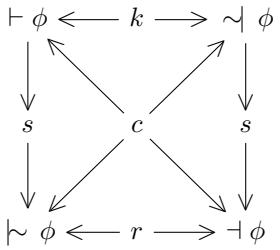
Proposition 3 *(Contraries) There is no ϕ s.t. $\vdash \phi$ and $\sim\!\!\dashv \phi$.*

Proof. Assume there is a ϕ such that $\vdash \phi$ and $\sim\!\!\dashv \phi$. By Proposition 1, it follows that $\mid\!\sim \phi$, but that contradicts the assumption that $\sim\!\!\dashv \phi$ by Corollary 2. ■

Proposition 4 *(Subcontraries) For all ϕ either $\mid\!\sim \phi$ or $\dashv \phi$.*

Proof. Assume it is not the case that for all ϕ either $\mid\!\sim \phi$ or $\dashv \phi$. Then there is ϕ s.t. $\sim\!\!\dashv \phi$ and $\vdash \phi$. Taking $\sim\!\!\dashv \phi$ it follows from Corollary 1 that $\dashv \phi$. By Proposition 2 we get a contradiction with $\vdash \phi$. ■

These propositions form the next square of opposition where c denotes contradictories, s subalterns, k contraries and r subcontraries.



Proposition 1 and Corollary 1 represent Supraclassicality; Proposition 2 and Corollary 2 stand for Consistency while the remaining statements specify the coherence of the square, and thus, the overall coherence of the system.

Consider, for example, a scenario in which an agent intends to acquire its PhD, and we set the next configuration Δ of beliefs and intentions: $F_B = \{\top\}$, $B = \{scolarship\}$, $F_I = \{research : \top \leftarrow \top\}$, $I = \{phd : \top \leftarrow thesis, exam; thesis : scolarship \leftarrow research; exam : \top \leftarrow research\}$. And suppose we send

the query: *phd?* The search of intentions with head *phd* in F_I fails, thus the alternative $\vdash \phi[phd]$ does not hold. Thus, we can infer, by contradiction rule (Proposition 2), that it is not strongly provable that *phd*, i.e., that eventually in some state the intention *phd* does not hold. Thus, the result of the query should be that the agent will get its PhD defeasibly under the Δ configuration. On the contrary, the query *research?* will succedd as $\vdash \phi[research]$, and thus, we would say *research* is both strongly and weakly provable (Proposition 1).

3.2 Soundness

The framework is Sound with respect to its semantics.

Definition 7 *(Satisfaction) A formula ϕ is true in K iff ϕ is true in all configurations σ in K. This is to say, $K \models \phi \Leftrightarrow K, \sigma \models \phi$ for all $\sigma \in S$.*

Definition 8 *(Run of an agent in a model) Given an initial configuration β, a transition system Γ and a valuation V, $K^{\beta}_{\Gamma} = \left\langle S^{\beta}_{\Gamma}, R^{\beta}_{\Gamma}, V \right\rangle$ denotes a run of an agent in a model.*

Definition 9 *(Validity) A formula $\phi \in BDI^{CTL}_{AS(L)}$ is true for any agent run in Γ iff $\forall K^{\beta}_{\Gamma} \models \phi$*

By denoting $(\exists K^{\beta}_{\Gamma} \models \phi \ \mathsf{U} \ \neg \mathsf{BEL}(ctx(\phi))) \vee \models \phi$ as $\models\!\!\approx \phi$, and assuming $\models \phi \geq \models\!\!\approx \phi$ and $\approx\!\!\models \phi \geq =\!\!\models \phi$, a series of *translations* can be found s.t.:

$$\vdash \phi \longrightarrow \forall K^{\beta}_{\Gamma} \models \phi \longrightarrow \models \phi$$
$$\searrow \qquad\qquad \downarrow$$
$$\vdash\!\sim \phi \longrightarrow \models\!\!\approx \phi$$

And also for the rest of the fragments:

$$\sim\!\!\dashv \phi \longrightarrow \exists K^{\beta}_{\Gamma} \models \neg\phi \wedge \forall K^{\beta}_{\Gamma} \models \neg(\phi \ \mathsf{U} \ \neg \mathsf{BEL}(ctx(\phi))) \longrightarrow \approx\!\!\dashv \phi$$
$$\searrow \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow$$
$$\dashv \phi \longrightarrow\!\!\dashv\!\sim \phi$$

Proposition 5 *The following relations hold:*

$$a)\ If \vdash \phi\ then \models \phi \qquad b)\ If \ \vdash\!\sim \phi\ then \models\!\!\approx \phi$$

Proof. Base case. Taking Δ_i as a sequence with $i = 1$.

Case a) If we assume $\vdash \phi$, we have two subcases. First subcase is given by Definition 6 item 1.1. Thus we have $\Box\mathsf{A}(\mathsf{INT}(\phi))$. This means, by Definition 5 items P4 and S5 and Definition 4, that for all paths and all states $\phi \in C_I \vee C_E$. We can

represent this expression, by way of a translation, in terms of runs. Since paths and states are sequences of agent configurations we have that $\forall K_\Gamma^\beta \models \phi$, which implies $\models \phi$. Second subcase is given by Definition 6 item 1.2, which in terms of runs means that for all runs $\exists \phi[g] \in F_I : \mathsf{BEL}(ctx(\phi)) \wedge \forall \psi[g'] \in body(\phi) \vdash \psi[g']$. Since Δ_1 is a single step, $body(\phi) = \top$ and for all runs $\mathsf{BEL}(ctx(\phi)))$, $ctx(\phi) \in F_B$. Then $\forall K_\Gamma^\beta \models \phi$ which, same as above, implies $\models \phi$.

Case b) Let us suppose $\mathrel{|\!\sim} \phi$. Then we have two subcases. The first one is given by Definition 6 item 2.1. So, we have that $\vdash \phi$ which, as we showed above, already implies $\models \phi$. On the other hand, by item 2.2, we have $\dashv \neg\phi$ and two alternatives. The first alternative, item 2.2.1, is $\Diamond \mathsf{E}(\mathsf{INT}(\phi)\ \mathsf{U}\ \neg\mathsf{BEL}(ctx(\phi)))$. Thus, we can reduce this expression by way of Definition 5 items P3 and S4, to a translation in terms of runs: $\exists K_\Gamma^\beta \models \phi\ \mathsf{U}\ \neg\mathsf{BEL}(ctx(\phi))$, which implies $\mathrel{|\!\approx} \phi$. The second alternative comes from item 2.2.2, $\Diamond \mathsf{E}(\exists \phi[g] \in I : \mathsf{BEL}(ctx(\phi)) \wedge \forall \psi[g'] \in body(\phi) \mathrel{|\!\sim} \psi[g'])$ which in terms of runs means that for some run $\exists \phi[g] \in I : \mathsf{BEL}(ctx(\phi)) \wedge \forall \psi[g'] \in body(\phi) \mathrel{|\!\sim} \psi[g']$, but Δ_1 is a single step, and thus $body(\phi) = \top$. Thus, there is a run in which $\exists \phi[g] \in I : \mathsf{BEL}(ctx(\phi))$, i.e., $(\exists K_\Gamma^\beta \models (\phi\ \mathsf{U}\ \neg\mathsf{BEL}(ctx(\phi))))$ by using the weak case of Definition 6 P5. Thus, by addition, $(\exists K_\Gamma^\beta \models (\phi\ \mathsf{U}\ \neg\mathsf{BEL}(ctx(\phi))))\vee \models \phi$, and therefore, $\mathrel{|\!\approx} \phi$.

Inductive case. Case a) Let us assume that for $n \leq k$, if $\Delta_n \vdash \phi$ then $\Delta \models \phi$. And suppose Δ_{n+1}. Further, suppose $\Delta_n \vdash \phi$, then we have two alternatives. First one being, by Definition 6 item 1.1, that we have an intention ϕ s.t. $ctx(\phi) = body(\phi) = \top$. Since $body(\phi)$ is empty, it trivially holds at n, and by the induction hypothesis, $body(\phi) \subseteq \Delta_{n+1}$, and thus $\models \phi$. Secondly, by Definition 6 item 1.2, for all runs $\exists \phi[g] \in I : \mathsf{BEL}(ctx(\phi)) \wedge \forall \psi[g'] \in body(\phi) \vdash \psi[g']$. Thus, for all runs n, $\forall \psi[g'] \in body(\phi) \vdash \psi[g']$, and so by the induction hypothesis, $body(\phi) \subseteq \Delta_{n+1}$, i.e., $\Delta \vdash \psi[g']$. Therefore, $\models \phi$.

Case b) Let us assume that for $n \leq k$, if $\Delta_n \mathrel{|\!\sim} \phi$ then $\Delta \mathrel{|\!\approx} \phi$. And suppose Δ_{n+1}. Assume $\Delta_n \mathrel{|\!\sim} \phi$. We have two alternatives. The first one is given by Definition 6 item 2.1, i.e., $\vdash \phi$, which already implies $\models \phi$. The second alternative is given by item 2.2, $\Delta \dashv \neg\phi$ and two subcases: $\Diamond \mathsf{E}(\mathsf{INT}(\phi)\ \mathsf{U}\ \neg\mathsf{BEL}(ctx(\phi)))$ or $\Diamond \mathsf{E}(\exists \phi[g] \in I : \mathsf{BEL}(ctx(\phi)) \wedge \forall \psi[g'] \in body(\phi) \mathrel{|\!\sim} \psi[g'])$. If we consider the first subcase there are runs n which comply with the definition of $\mathrel{|\!\approx} \phi$. In the remaining subcase we have $\forall \psi[g'] \in body(\phi) \mathrel{|\!\sim} \psi[g']$, since $body(\phi) \subseteq \Delta_n$, by the induction hypothesis $\Delta \mathrel{|\!\sim} \psi[g']$, and thus, $\Delta_{n+1} \mathrel{|\!\sim} \phi$, i.e., $\mathrel{|\!\approx} \phi$. ■

Corollary 3 *The following relations hold:*

$$a)\ If\ \dashv \phi\ then\ =\!|\ \phi \qquad b)\ If\ \sim\!|\ \phi\ then\ \approx\!|\ \phi$$

3.3 Other formal properties

But there are other formal properties that may be used to explore and define the rationality of intentional reasoning, i.e., its good behavior. In first place, it results quite reasonable to impose Reflexivity on the consequence relation so that if $\phi \in \Delta$, then $\Delta \mathrel{|\!\sim} \phi$.

Further, another reasonable property should be one that dictates that strong intentions imply weak intentions. In more specific terms, that if an intention ϕ follows from Δ in a monotonic way, then it must also follow according to a non-monotonic approach. Thus, in second place, we need the reasonable requirement that intentions strongly mantained have to be also weakly mantained, but no the other way around:

Proposition 6 *(Supraclassicality) If* $\Delta \vdash \phi$*, then* $\Delta \mid\sim \phi$*.*

Proof. See Proposition 1. ■

Another property, a very strong one, is Consistency Preservation. This property tells us that if some intentional set is classically consistent, then so is the set of defeasible consequences of it.

Proposition 7 *(Consistency preservation) If* $\Delta \mid\sim \bot$*, then* $\Delta \vdash \bot$*.*

Proof. Let us consider the form of the intention $\bot$. Such intention is the intention of the form $\phi \wedge \neg\phi$, which is, therefore, impossible to achieve, that is to say, for all agent runs, $\mid\sim \bot$ is never achieved. Thus $\Delta \mid\sim \bot$ is false, which makes the whole implication true. ■

And, if an intention ϕ is a consequence of Δ, then ψ is a consequence of Δ and ϕ only if it is already a consequence of Δ, because adding to Δ some intentions that are already a consequence of Δ does not lead to any *increase* of information. In terms of the size of a proof [1], such size does not affect the degree to which the initial information supports the conclusion:

Proposition 8 *(Cautious cut) If* $\Delta \mid\sim \phi$ *and* $\Delta, \phi \mid\sim \psi$ *then* $\Delta \mid\sim \psi$*.*

Proof. Let us start by transforming the original proposition into the next one: if $\Delta \nmid\sim \psi$ then it is not the case that $\Delta \mid\sim \phi$ and $\Delta, \phi \mid\sim \psi$. Further, this proposition can be transformed again: if $\Delta \nmid\sim \psi$ then either $\Delta \nmid\sim \phi$ or $\Delta, \phi \nmid\sim \psi$ from which, using Corollary 1, we can infer: if $\Delta \dashv \psi$ then either $\Delta \dashv \phi$ or $\Delta, \phi \dashv \psi$. Now, let us assume that $\Delta \dashv \psi$ but it is not the case that either $\Delta \dashv \phi$ or $\Delta, \phi \dashv \psi$, i.e., that $\Delta \dashv \psi$ but $\Delta \vdash \phi$ and $\Delta, \phi \vdash \psi$. Considering the expression $\Delta, \phi \vdash \psi$ we have two alternatives: either $\psi \in \mathit{body}(\phi)$ or $\psi \notin \mathit{body}(\phi)$. In the first case, given that $\Delta \vdash \phi$ then, since $\psi \in \mathit{body}(\phi)$ it follows that $\vdash \psi$, but that contradicts the assumption that $\Delta \dashv \psi$. In the remaining case, if $\Delta, \phi \vdash \psi$ but $\psi \notin \mathit{body}(\phi)$, then $\Delta \vdash \psi$, which contradicts the assumption that $\Delta \dashv \psi$. ■

If we go a little bit further, we should look for some form of Cautious Monotony as the converse of Cut in such a way that if ϕ is taken back into Δ that does not lead to any *decrease* of information, that is to say, that adding implicit information is a monotonic task:

Proposition 9 *(Cautious monotony) If* $\Delta \mid\sim \psi$ *and* $\Delta \mid\sim \gamma$ *then* $\Delta, \psi \mid\sim \gamma$*.*

Proof. Let us transform the original proposition: if $\Delta, \psi \nmid\sim \gamma$ then it is not the case that $\Delta \mid\sim \psi$ and $\Delta \mid\sim \gamma$. Thus, if $\Delta, \psi \nmid\sim \gamma$ then either $\Delta \nmid\sim \psi$ or

$\Delta \sim\!\!\!\!\mid \gamma$, and by Corollary 1, if $\Delta, \psi \dashv \gamma$ then either $\Delta \dashv \psi$ or $\Delta \dashv \gamma$. Now, let us suppose that $\Delta, \psi \dashv \gamma$ but it is false that either $\Delta \dashv \psi$ or $\Delta \dashv \gamma$, this is to say, that $\Delta, \psi \dashv \gamma$ and $\Delta \vdash \psi$ and $\Delta \vdash \gamma$. Regarding the expression $\Delta, \psi \dashv \gamma$ we have two alternatives: either $\gamma \in body(\psi)$ or $\gamma \notin body(\psi)$. In the first case, since $\gamma \in body(\psi)$ and $\Delta \dashv \psi$, then $\dashv \gamma$, which contradicts the assumption that $\Delta \vdash \gamma$. On the other hand, if we consider the second alternative, $\Delta \dashv \gamma$, but that contradicts the assumption that $\Delta \vdash \gamma$. ■

4 Conclusion

It seems reasonable to conclude that this bratmanian model of intentional reasoning captures relevant features of the nature of intentional reasoning and can be modelled in a well-behaved defeasible logic that clarifies its status, since it satisfies conditions of Consistency, Soundness, Supraclassicality, Reflexivity, Consistency Preservation, Cautious Cut and Cautious Monotony. In other words, it is plausible to conclude that intentional reasoning has the right to be called *logical reasoning* since it behaves, *mutatis mutandis*, as a logic, strictly speaking, as a non-monotonic logic.

The relevance of this work becomes clear once we notice that, although intentions have received a lot of attention, their dynamic features have not been studied completely [16]. There are formal theories of intentional reasoning [7,17,22,24] but very few of them consider the revision of intentions [16] or the non-monotonicity of intentions [13] as legitimate research topics, which we find odd since the foundational theory guarantees that such research is legitimate and necessary [4]. Recent works confirm the status of this emerging area [13,16,19].

Acknowledgements. The author would like to thank the anonymous reviewers for their helpful comments and precise corrections; and the School of Philosophy at UPAEP for all the assistance. This work has also been supported by the CONACyT scholarship 214783.

References

1. Antonelli, A.: Grounded Consequence for Defeasible Logic. Cambridge: Cambridge University Press (2005)
2. Bordini, R.H., Moreira, Á.F.: Proving BDI properties of agent-oriented programming languages. Annals of Mathematics and Artificial Intelligence 42, 197–226 (2004)
3. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley, England (2007)
4. Bratman, M.: Intention, Plans, and Practical Reason. Harvard University Press, Cambridge (1987)
5. Castro-Manzano, J.M., Barceló-Aspeitia. A.A. and Guerra-Hernández, A.: Consistency and soundness for a defeasible logic of intention. Advances in soft computing algorithms, Research in Computing Science vol. 54, (2011)

6. Clarke, E. M. Jr., Grumberg, O. and Peled. D. A.: Model Checking. MIT Press, Boston, MA., USA, (1999)
7. Cohen, P., Levesque, H.: Intention is choice with commitment. Artificial Intelligence 42(3), 213-261 (1990)
8. Couturat, L.: La logique de Leibniz d'aprés de documents inédits. G. Olms, Hildesheim (1962)
9. Dastani, M., van Riemsdijk, M.B., Meyer, J.C.: A grounded specification language for agent programs. In: AAMAS'07. ACM, New York, NY, pp. 1-8 (2007)
10. Emerson, A.: Temporal and modal logic. In: Handbook of Theoretical Computer Science, Elsevier Science Publishers B.V., Amsterdam, (1990)
11. Dear Bertrand Russell... A Selection of his Correspondence with the General Public 19501968, edited by Barry Feinberg and Ronald Kasrils. London, George Allen and Unwin, (1969)
12. Gabbay, D. M.: Theoretical foundations for nonmonotonic reasoning in expert systems. in K. Apt (ed.), Logics and Models of Concurrent Systems, Berlin and New York: Springer Verlag, pp. 439459 (1985).
13. Governatori, G., Padmanabhan, V. and Sattar, A.: A Defeasible Logic of Policy-based Intentions. In AI 2002: Advances in Artificial Intelligence, LNAI-2557. Springer Verlag (2002)
14. Governatori, G., Terenziani, P.: Temporal Extensions to Defeasible Logic. In Proceedings of IJCAI'07 Workshop on Spatial and Temporal Reasoning, India (2007)
15. A. Guerra-Hernández, J. M. Castro-Manzano, A. El-Fallah-Seghrouchni.: CTLAgentSpeak(L): a Specification Language for Agent Programs. Journal of Algorithms in Cognition, Informatics and Logic, (2009)
16. Hoek, W. van der, Jamroga, W., Wooldridge, M.: Towards a theory of intention revision. Synthese, Springer-Verlag (2007).
17. Konolige, K., Pollack, M. E.: A representationalist theory of intentions. In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-93), 390-395, San Mateo: Morgan Kaufmann (1993).
18. Nute, D.: Defeasible logic. In: INAP 2001, LNAI 2543M 151-169, Springer-Verlag, (2003).
19. Icard, Th., Pacuit. E., Shoham, Y.: Joint revision of belief and intention. Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning, (2010).
20. Prakken, H., Vreeswijk, G.: Logics for defeasible argumentation. In D. Gabbay and F. Guenthner (eds.), Handbook of Philosophical Logic, second edition, Vol 4, pp. 219-318. Kluwer Academic Publishers, Dordrecht etc., (2002).
21. Rao, A.S., Georgeff, M.P.: Modelling Rational Agents within a BDI-Architecture. In: Huhns, M.N., Singh, M.P., (eds.) Readings in Agents, pp. 317–328. Morgan Kaufmann (1998).
22. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: de Velde, W.V., Perram, J.W. (eds.) MAAMAW. LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg (1996).
23. Turner, R. and Eden, A.: "The Philosophy of Computer Science", The Stanford Encyclopedia of Philosophy (Summer 2009 Edition), Edward N. Zalta (ed.), URL = ¡http://plato.stanford.edu/archives/sum2009/entries/computer-science/¿.
24. Wooldridge, M.: Reasoning about Rational Agents. MIT Press, Cambridge (2000).

Appendix

AgentSpeak(L) syntax An agent ag is formed by a set of plans ps and beliefs bs (grounded literals). Each plan has the form $te : ctx \leftarrow h$. The context ctx of a plan is a literal or a conjunction of them. A non empty plan body h is a finite sequence of actions $A(t_1, \ldots, t_n)$, goals g (achieve ! or test ? an atomic formula $P(t_1, \ldots, t_n)$), or beliefs updates u (addition + or deletion −). $\top$ denotes empty elements, e.g., plan bodies, contexts, intentions. The trigger events te are updates (addition or deletion) of beliefs or goals. The syntax is shown in Table 1.

ag	$::= bs \quad ps$	h	$::= h_1; \top \mid \top$
bs	$::= b_1 \ldots b_n \ (n \geq 0)$	h_1	$::= a \mid g \mid u \mid h_1; h_1$
ps	$::= p_1 \ldots p_n \ (n \geq 1)$	at	$::= P(t_1, \ldots, t_n) \ (n \geq 0)$
p	$::= te : ctx \leftarrow h$	a	$::= A(t_1, \ldots, t_n) \ (n \geq 0)$
te	$::= +at \mid -at \mid +g \mid -g$	g	$::= !at \mid ?at$
ctx	$::= ctx_1 \mid \top$	u	$::= +b \mid -b$
ctx_1	$::= at \mid \neg at \mid ctx_1 \wedge ctx_1$		

Table 1. Sintax of $AgentSpeak(L)$.

AgentSpeak(L) semantics The operational semantics of $AgentSpeak(L)$ are defined by a transition system, as showed in Figure 1, between configurations $\langle ag, C, M, T, s \rangle$:

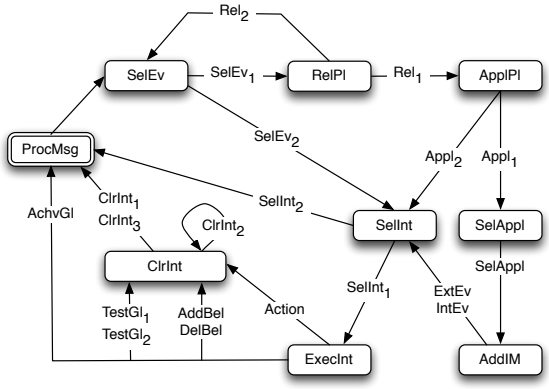


Fig. 1. The interpreter for $AgentSpeak(L)$ as a transition system.

Under such semantics a run is a set $Run = \{(\sigma_i, \sigma_j) | \Gamma \vdash \sigma_i \rightarrow \sigma_j\}$ where Γ is the transition system defined by the $AgentSpeak(L)$ operational semantics and σ_i, σ_j are agent configurations.

A Mobility Logic for Object Net Systems

Frank Heitmann and Michael Köhler-Bußmeier

University of Hamburg, Department for Informatics
Vogt-Kölln-Straße 30, D-22527 Hamburg
{heitmann,koehler}@informatik.uni-hamburg.de

Abstract. In this paper we present work in progress on a special variant of *Object Petri Nets* and on the introduction of a *Mobility Logic* to reason about them.
The Petri nets considered in this paper allow the *vertical transport* of net tokens i.e. the transport of net tokens through different nesting levels, giving one enhanced modelling capabilities and allowing one to naturally model certain situations arising in nested structures.
The logic then allows us not only to reason about the evolution of the described system in time, but also about spatial configurations, i.e. in this logic we can express for example, that a certain object or agent is always somewhere or at a specific location. This part of our work is inspired by the work of Cardelli and Gordon on the Ambient Calculus and the Ambient Logic.

Keywords: design methods, higher-level Petri net models, nets-within-nets, mobility logic, model checking and verification

1 Introduction

Object Petri Nets are Petri Nets whose tokens may be Petri Nets again and thus may have an inner structure and activity. This approach is useful to model mobile systems and other systems arising in Computer Science which enjoy a certain nesting of structures (cf. [10] and [11]).

This approach, which is also called the *nets-within-nets* paradigm, was proposed by Valk [22, 23] for a two levelled structure and generalised in [12, 13] for arbitrary nesting structures. By now many related approaches like recursive nets [6], nested nets [18], adaptive workflow nets [19], AHO systems [9], PN^2 [8], Mobile Systems [17], and many others are known. See [14] for a detailed discussion.[1]

A variant introduced a few years ago in [15], allows the *vertical transport* of net tokens, i.e. the transport of net tokens through different nesting levels, giving one enhanced modelling capabilities and allowing one to naturally model certain situations arising in nested structures.

[1] Another line of research also dealing with nesting, but not in the field of Petri nets, is concerned with process calculi. Arguably most prominently there are the Ambient Calculus of Gordon and Cardelli [2] and the Seal Calculus [3] among many others.

Unfortunately the formalism was rather complicated and thus not well suited for neither theoretical investigations nor modelling applications. In the first part of this paper we devise a more convenient variant with regard to theoretical investigations than the variant known so far. The variant proposed here retains the ability to transport tokens in the vertical dimension, but in particular restricts the transitions participating in the firing to at most two levels.

After introducing the formalism we go on and introduce a logic that allows us not only to reason about the evolution of the described system in time, but also about spatial configurations, i.e. a logic in which we can express for example, that a certain object or agent is always somewhere or at a specific location. This part of our work is deeply inspired by the work of Cardelli and Gordon on the Ambient Calculus and the Ambient Logic [2], [1].

While the main part of this presentation deals with the introduction of the formalism and the logic and thus with definitions and examples, we also hint at work in progress regarding the complexity of certain problems for object nets and the newly developed logic. In particular we show that the reachability problem is decidable in PSPACE for a specific finite-state-segment of our formalism and argue that the model checking problem for the new logic might also be in PSPACE for this variant.

In the following we assume basic knowledge of Petri nets, see e.g. [20].

2 Object Nets

In [15] we presented a formalism for object nets which was rather complicated. The firing rule was particularly hard to formulate and to understand as were the events themselves. Unsurprisingly the formalism was Turing-complete, but many of the formalism's facets where not even used in the proof.

In the following we will present a stripped-down variant that still captures the essentials of the formalism in [15], namely the nets-within-nets structure, the synchronisation, and in particular the possibility to transport nets *vertically* through the channels. For an example take a look at Figure 3. An object net resides on place p' whose place p is again marked by another object net. The transitions t' and t use the same channel descriptor c and the channel properties match.[2] Ignoring the inner structure of the net tokens both transitions are activated and may fire. The successor marking is pictured in Figure 4. The net token previously on p' has travelled to p''', but it's place p is now empty, because that object net has travelled in the vertical dimension via channel c to the place p''. In the following we will give a formal description of this formalism.

An *Object Net System* (ONS for short) consist of a *system net* $\widehat{N} = (\widehat{P}, \widehat{T}, \widehat{F})$ and a finite number of *object nets* $\mathcal{N} = \{N_1, \ldots, N_m\}$, $N_i = (P_i, T_i, F_i)$. Black tokens can be described by a special object net which has no places and transitions. We set $\widehat{\mathcal{N}} := \mathcal{N} \cup \widehat{N}$. Instead of P_i, T_i and so on we sometimes make use

[2] This will be defined later, for now note that the channel property $\uparrow_{N_1}$ of t means that t wants to send a object of type N_1 *upwards* and the channel property $\cap_{N_1}$ of t' means that it wants to *catch* a object of type N_1 (both via channel c).

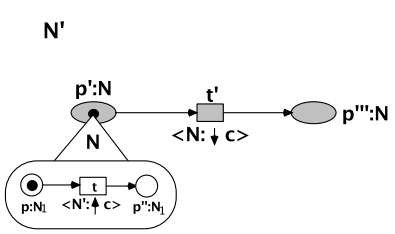


Fig. 1. Before Firing.

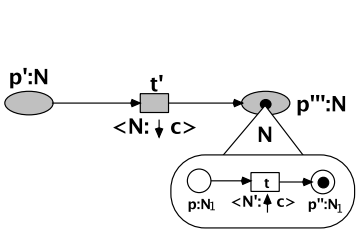


Fig. 2. After Firing.

of the notation $T(N_i) := T_i$, i.e. given an object net N the set of its transitions is denoted by $T(N)$, the set of its places by $P(N)$. We use $\mathcal{P}_\mathcal{N} := \dot{\bigcup}_{N\in\mathcal{N}} P(N)$, $\mathcal{P} := \mathcal{P}_\mathcal{N} \cup \widehat{P}$, $\mathcal{T}_\mathcal{N} := \dot{\bigcup}_{N\in\mathcal{N}} T(N)$, and $\mathcal{T} := \mathcal{T}_\mathcal{N} \cup \widehat{T}$ to denote the set of all places and transitions.

The places are all typed via the *typing function* $d : \mathcal{P} \to \mathcal{N}$. Note that no place is typed with the system net $\widehat{N}$.

Transitions are labelled with channels to allow for synchronisation. Channels consist of a *descriptor* taken from a finite set of *channel descriptors* $C_d = \{c_1, c_2, \ldots, c_n\}$ and a *channel property* $C_p = \{\Uparrow, \Downarrow, \Uparrow_{N_1}, \ldots, \Uparrow_{N_m}, \Downarrow_{N_1}, \ldots, \Downarrow_{N_m}, \cup_{N_1}, \ldots, \cup_{N_m}, \cap_{N_1}, \ldots, \cap_{N_m}\}$. A *channel* is then a element of the set $C := C_p \times C_d$, where instead of e.g. $(\Uparrow, c_1)$ we usually simply write $\Uparrow c_1$.

Since the system net is at the highest level of the hierarchy, not every channel can be used there. To ease the notation later we additionally define $\widehat{C} := (C_p \setminus \{\Uparrow, \Uparrow_{N_1}, \ldots, \Uparrow_{N_m}, \cup_{N_1}, \ldots, \cup_{N_m}\}) \times C_d$.

The *labelling functions* are now defined as

$$\widehat{l} : \widehat{T} \to (\widehat{C} \times \mathcal{N}) \cup \{\epsilon\}$$

and for each $i \in [m]$ as

$$l_i : T_i \to (C \times \widehat{\mathcal{N}}) \cup \{\epsilon\}$$

which are combined to

$$l : \mathcal{T} \to (C \times \widehat{\mathcal{N}}) \cup \{\epsilon\}$$

with $l(t) = \widehat{l}(t)$ if $t \in \widehat{T}$ and $l(t) = l_i(t)$ if $t \in T_i$.

Note that each transition is labelled with exactly one channel or ϵ. The intended meaning of $l(t) = (c, N)$ is that t synchronizes via channel c with a net of type N. In the case of $l(t) = \epsilon$ the transition t fires autonomously.

We now describe the possible labellings together with there intended meaning and the restrictions the labellings impose on the nets' structure.

1. $l(t) = \epsilon$, $t \in T(N)$. In this case there is no synchronisation and t fires in principal as in a normal p/t net.

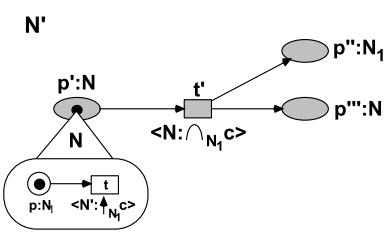


Fig. 3. Before Firing.

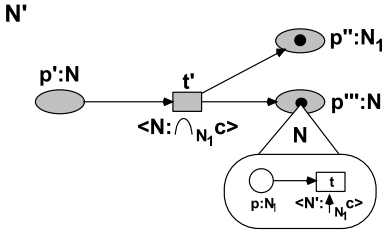


Fig. 4. After Firing.

2. $l(t) = (\Uparrow\ c, N')$, $t \in T(N), N \neq \widehat{N}$[3]. The labelling means that t wants to synchronize (via c) with a transition in N', where N' is a net *"above"* N, i.e. N is a net-token in N' (see Figure 1 and 2). Formally we demand a place p' in N' with $d(p') = N$ and a transition $t' \in p'^{\bullet}$ with $l(t') = (\Downarrow c, N)$.
3. $l(t') = (\Downarrow c, N)$, $t' \in T(N')$. The complement to the above case. There is now a place $p' \in {}^{\bullet}t'$ with $d(p') = N$ and in N there is a transition t with $l(t) = (\Uparrow c, N')$.
4. $l(t) = (\Uparrow_{N_1} c, N'), t \in T(N), N \neq \widehat{N}$. Similar to $\Uparrow$ above, t wants to synchronize (via c) with a transition in N' "above". This time additionally a net of type N_1 is send from N through c upwards to N' (resp. to a place in the postset of the transition in N' that uses the channel c). The situation is depicted in Figures 3 and 4. Note that the token on place p (Fig. 3) resp. place p'' (Fig. 4) can be an object net. Formally there is a place p' with $d(p') = N$ and a transition $t' \in p'^{\bullet}$ with $l(t') = (\cap_{N_1} c, N)$.[4] Moreover there is a place $p \in {}^{\bullet}t$ with $d(p) = N_1$ and *no place* in the postset of t of this type. In N' there is a place $p'' \in t'^{\bullet}$ with $d(p'') = N_1$ and *no place* in the preset of t' of this type. (The net of type N_1 thus travels from p (in N) to p'' (in N').)
5. $l(t') = (\cap_{N_1} c, N)$. The complement to the case above, but similar to $\Downarrow$. There is a place $p' \in {}^{\bullet}t'$ with $d(p') = N$ and in N there is a transition t with $l(t) = (\Uparrow_{N_1} c, N')$. Moreover there is a place $p \in t^{\bullet}$ with $d(p) = N_1$ and *no place* in the postset of t with this type, and also a place $p'' \in t'^{\bullet}$ in N' with $d(p'') = N_1$ and *no place* in the preset of t of this type.
6. $l(t) = (\Downarrow_{N_1} c, N'), t \in T(N)$. Similar to $\Downarrow$ above, t wants to synchronize via c with a transition in N' "below". This time a net of type N_1 is additionally send from N through c downwards to N' (resp. to a place in the postset of the transition in N' that uses the channel $\cup_{N_1} c$). The situation is depicted in Figures 5 and 6. Note again that the token on place p (Fig. 5) resp. place p'' (Fig. 6) can be an object net. Formally there is a place $p \in {}^{\bullet}t$ with $d(p) = N'$, a transition t' in N' with $l(t') = (\cup_{N_1} c, N)$ and moreover a place $p' \in {}^{\bullet}t$ with $d(p') = N_1$ and a place $p'' \in t'^{\bullet}$ with $d(p'') = N_1$. There is no

[3] In the system net $\widehat{N}$ the channel property $\Uparrow$ can not be used.
[4] The usage of the symbol $\cap$ shall illustrate that a net coming from below is "caught.

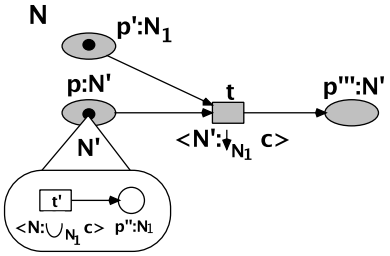


Fig. 5. Before Firing.

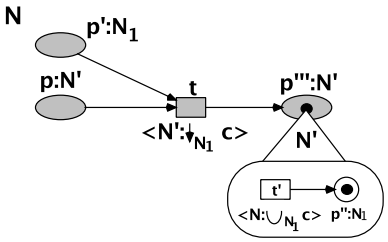


Fig. 6. After Firing.

place in the postset of t or in the preset of t' of type N_1. (The Net of type N_1 thus travels from p' (in N) to p'' (in N').)

7. $l(t') = (\cup_{N_1} c, N)$, $t' \in T(N'), N' \neq \widehat{N}$. Again the complement to the case directly above.

In addition to the above described restrictions on the nets' structure imposed by the labelling, we demand that each type appears at most once in the preset and in the postset of a transition, i.e.

$$\forall N \in \widehat{\mathcal{N}}\ \forall t \in T(N) : |\{p \mid p \in {}^\bullet t \wedge d(p) = N\}|, |\{p \mid p \in t^\bullet \wedge d(p) = N\}| \leq 1$$

Markings. To define markings, which will turn out to be nested multi-sets, let OS be an object net system as above consisting of a system net $\widehat{N}$ and a finite set of object nets $\mathcal{N}$. Furthermore let $d : \mathcal{P} \to \mathcal{N}$ be the typing function (no place is typed with the system net). Now let

$$\mathcal{M}_0(N) := \{p[\mathbf{0}] \mid p \in P(N)\}$$

for a $N \in \widehat{\mathcal{N}}$ and let $\mathcal{M}_0 := \cup_{N \in \widehat{\mathcal{N}}} \mathcal{M}_0(N)$. Note that with $\mu \in \mathcal{MS}(\mathcal{M}_0(N))$ for a fix $N \in \widehat{\mathcal{N}}$ we can describe how many *empty* net tokens reside on each place of N (this includes black tokens, which are just special net tokens in our setting). The multiset μ is thus similar to the usual multiset of places that describes a marking.

Let

$$\mathcal{M}_{i+1}(N) := \{p[\mu] \mid p \in P(N) \wedge \mu \in \mathcal{MS}(\cup_{k \leq i} \mathcal{M}_k(d(p))\} \text{ and}$$
$$\mathcal{M}_{i+1} := \cup_{N \in \widehat{\mathcal{N}}} \mathcal{M}_{i+1}(N)$$

and finally let

$$\mathcal{M} := \mathcal{MS}(\cup_{i \geq 0} \mathcal{M}_i) = \mathcal{MS}(\cup_{i \geq 0} \cup_{N \in \widehat{\mathcal{N}}} \mathcal{M}_i(N)).$$

Each *nested multiset* $\mu \in \mathcal{M}, \mu = \sum_{k=1}^{n} \widehat{p}_k[M_k]$, is a *marking* of the object net system OS, where $\widehat{p}_k$ is a place in the system net and M_k is a marking of a net-token of type $d(\widehat{p}_k)$, which again might be a nested multiset.

We extend addition, subtraction and $\leq$-relations etc. for nested multisets in the usual way, e.g. $\mu \leq \mu'$ for two nested multisets if another nested multiset ρ exists such that $\mu+\rho = \mu'$. Furthermore, we need a relation to address the nesting of markings. We write $\mu \bigtriangledown \mu'$ to indicate that the submarking μ' is contained in the marking μ within exactly one level of nesting:

$$\mu \bigtriangledown \mu' \quad \text{iff} \quad \exists p \in \mathcal{P}, \mu'' \in \mathcal{M} \,.\, \mu \equiv p[\mu'] + \mu''$$

The reflexive and transitive closure of this relation is denoted by $\bigtriangledown^*$ as usual. Thus $\mu \bigtriangledown^* \mu'$ means that μ contains μ' at some nesting level.

Note that $\mathcal{M}$ differs for different object net systems. If necessary we will denote the set of possible markings of a ONS OS by $\mathcal{M}_{OS}$, but if no ambiguities can arise, we neglect the subscript.

Given a (sub-)marking μ we use $\Pi^1(\mu)$ to abstract away the substructure of all net-tokens and $\Pi^2_N(\mu)$ for the summed up marking of all net tokens of type $N \in \mathcal{N}$ ignoring their local distribution, i.e.

$$\Pi^1(\sum_{k=1}^{n} p_k[M_k]) = \sum_{k=1}^{n} p_k$$
$$\Pi^2_N(\sum_{k=1}^{n} p_k[M_k]) = \sum_{k=1}^{n} \mathbf{1}_N(p_k) \cdot M_k,$$

where $\mathbf{1}_N : \mathcal{P} \to \{0,1\}$ with $\mathbf{1}_N(p) = 1$ iff $d(p) = N$. Note that the summation in Π^2_N is *not* recursive, i.e. a marking of a net token of type N on a deeper nesting level is not summed up (but remains in the sub-marking M_k). Defined in this way Π^2_N is useful to describe the firing rule.

Object Net Systems, Events, and the Firing Rule.

Definition 1. *An Object Net System (ONS) is a tuple $OS = (\widehat{N}, \mathcal{N}, d, l)$ with*

1. *The system net $\widehat{N}$,*
2. *a finite set of object nets $\mathcal{N}$,*
3. *the typing function $d : \mathcal{P} \to \mathcal{N}$, and*
4. *the labelling function $l : \mathcal{T} \to (C \times \widehat{\mathcal{N}}) \cup \{\epsilon\}$, which is consistent with the structural restrictions mentioned above.*

An ONS with initial marking is a tuple $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ where the initial marking $\mu_0 \in \mathcal{M}$ is a marking of $\widehat{N}$, i.e. there is a k such that $\mu_0 \in \mathcal{M}_k(\widehat{N})$.

To define *events* and the *firing rule* we distinguish four cases in accordance with the labelling above:

1. $(t, t') \in \mathcal{T} \times \mathcal{T}$ with $l(t) = (\Uparrow_{N_1} c, N')$ and $l(t') = (\cap_{N_1} c, N)$ (Fig. 3 and 4).
2. $(t, t') \in \mathcal{T} \times \mathcal{T}$ with $l(t) = (\Downarrow_{N_1} c, N')$ and $l(t') = (\cup_{N_1} c, N)$ (Fig. 5 and 6).
3. $(t, t') \in \mathcal{T} \times \mathcal{T}$ with $l(t) = (\Uparrow c, N')$ and $l(t') = (\Downarrow c, N)$ (Fig. 1 and 2).
4. $t \in \mathcal{T}$ with $l(t) = \epsilon$.

The first three cases are *synchronous* events, the last one describes an *autonomous* event.

Now for the first case let μ be the current marking and let $\lambda, \lambda', \rho, \rho' \leq \mu$ be sub-markings with $\lambda' \leq \lambda$ and $\rho' \leq \rho$. The intended meaning is that λ is the sub-marking of μ enabling t' and λ' is the sub-marking (of λ) that enables t in the synchronous event. Then ρ is the resulting sub-marking with regard to t' and ρ' with regard to t. Furthermore a net of type N_1 is removed from λ' and added to ρ.

This is expressed in the firing predicate $\phi_{\Uparrow_{N_1},\cap_{N_1}}$:

$$\begin{aligned}
&\phi_{\Uparrow_{N_1},\cap_{N_1}}(t,t',\lambda,\lambda',\rho,\rho') \iff \\
&\qquad \Pi^1(\lambda) = \mathbf{pre}(t') \wedge \Pi^1(\rho) = \mathbf{post}(t') \wedge \\
&\qquad \Pi^1(\lambda') = \mathbf{pre}(t) \wedge \Pi^1(\rho') = \mathbf{post}(t) \wedge \\
&\qquad \forall N' \in \mathcal{N} \setminus \{N, N_1\} : \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) \wedge \\
&\qquad \forall N' \in \mathcal{N} \setminus \{N_1\} : \Pi^2_{N'}(\rho') = \Pi^2_{N'}(\lambda') \wedge \\
&\qquad \Pi^2_{N}(\rho) = \Pi^2_{N}(\lambda) - \lambda' + \rho' \wedge \\
&\qquad \Pi^2_{N_1}(\rho) = \Pi^2_{N_1}(\lambda') \wedge \\
&\qquad \Pi^2_{N_1}(\rho') = \mathbf{0}
\end{aligned} \tag{1}$$

The first two lines take care of activation of t and t' and the correct successor marking. Lines 3 and 4 handle non involved object nets and the last three lines correctly relate the different (sub-)markings with regard to the synchronous event, i.e. with regard to the two firing transitions.

The other cases are quite similar and the third and fourth case can even be seen as special (and easier) cases to the above.

For the second case the firing predicate is given by

$$\begin{aligned}
&\phi_{\Downarrow_{N_1},\cup_{N_1}}(t,t',\lambda,\lambda',\rho,\rho') \iff \\
&\qquad \Pi^1(\lambda) = \mathbf{pre}(t) \wedge \Pi^1(\rho) = \mathbf{post}(t) \wedge \\
&\qquad \Pi^1(\lambda') = \mathbf{pre}(t') \wedge \Pi^1(\rho') = \mathbf{post}(t') \wedge \\
&\qquad \forall N' \in \mathcal{N} \setminus \{N, N_1\} : \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) \wedge \\
&\qquad \forall N' \in \mathcal{N} \setminus \{N_1\} : \Pi^2_{N'}(\rho') = \Pi^2_{N'}(\lambda') \wedge \\
&\qquad \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) - \lambda' + \rho' \wedge \\
&\qquad \Pi^2_{N_1}(\rho) = \mathbf{0} \wedge \\
&\qquad \Pi^2_{N_1}(\rho') = \Pi^2_{N_1}(\lambda)
\end{aligned} \tag{2}$$

Note that λ now enables t, λ' enables t', ρ is the resulting sub-marking with regard to t and ρ' with regard to t'. Furthermore a net of type N_1 is removed from λ and added to ρ' (and also to ρ, since $\rho' \leq \rho$). In principle the first two cases only differ in the last three lines that relate the different (sub-)markings and the firing transitions.

At last the third and fourth case:

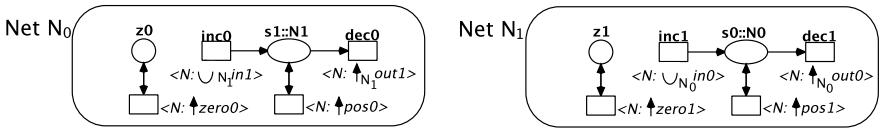


Fig. 7. The object nets N_0 and N_1 (from Theorem 1).

$$\begin{aligned}\phi_{\Uparrow,\Downarrow}(t,t',\lambda,\lambda',\rho,\rho') \iff\\ &\Pi^1(\lambda) = \mathbf{pre}(t') \wedge \Pi^1(\rho) = \mathbf{post}(t') \wedge\\ &\Pi^1(\lambda') = \mathbf{pre}(t) \wedge \Pi^1(\rho') = \mathbf{post}(t) \wedge\\ &\forall N \in \mathcal{N} \setminus \{N'\} : \Pi^2_N(\rho) = \Pi^2_N(\lambda) \wedge\\ &\Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) - \lambda' + \rho'\end{aligned} \tag{3}$$

$$\begin{aligned}\phi_\epsilon(t,\lambda,\rho) \iff\\ &\Pi^1(\lambda) = \mathbf{pre}(t) \wedge \Pi^1(\rho) = \mathbf{post}(t) \wedge\\ &\forall N \in \mathcal{N} : \Pi^2_N(\rho) = \Pi^2_N(\lambda)\end{aligned} \tag{4}$$

Note that the four firing predicates might a first glance look cumbersome, but are quite similar and in particular restrict every firing to two levels, which is far better tractable from a theoretical point of view than the firing rule introduced in [15] where a tree of synchronous transitions was able to fire. The firing rule can now be stated as follows:

Definition 2 (Firing Rule). *Let OS be an* ONS *and $\mu, \mu' \in \mathcal{M}$ markings. The synchronous event (t, t') is* enabled in μ for the mode $(\lambda, \lambda', \rho, \rho') \in \mathcal{M}^4$ *iff $\lambda' \leq \lambda \leq \mu$, $\rho' \leq \rho$ and one of $\phi_{\Uparrow_{N_1},\cap_{N_1}}$, $\phi_{\Downarrow_{N_1},\cup_{N_1}}$, or $\phi_{\Uparrow,\Downarrow}$ holds for $(t, t', \lambda, \lambda', \rho, \rho')$, according to the labelling of t and t'.*

An autonomous event $t, l(t) = \epsilon$ is enabled in μ for the mode (λ, ρ) iff $\lambda \leq \mu$ and ϕ_ϵ holds.

An event ϑ that is enabled in μ for a mode can fire: $\mu \xrightarrow[OS]{\vartheta} \mu'$. The resulting successor marking is defined as $\mu' = \mu - \lambda + \rho$.

The set of events is denoted by Θ. Firing is extend to sequences $w \in \Theta*$ in the usual way. The set of reachable markings from a marking μ is denoted by $RS_{OS}(\mu)$ or simply $RS(\mu)$. The reachability problem asks given an ONS *OS with initial marking μ_0 and a marking μ, if $\mu \in RS_{OS}(\mu_0)$ holds.*

2.1 Turing-Completeness of Object Net Systems

In [15] we have shown that the there defined object net formalism can directly simulate counter programs and thus is Turing-complete. We have severely restricted the formalism here, but retained the general ability to transfer net-tokens in the vertical dimension of the nested marking. The formalism devised here remains Turing-complete and indeed the proof in [15] can be easily adjusted to our new setting. We will only sketch the proof here.

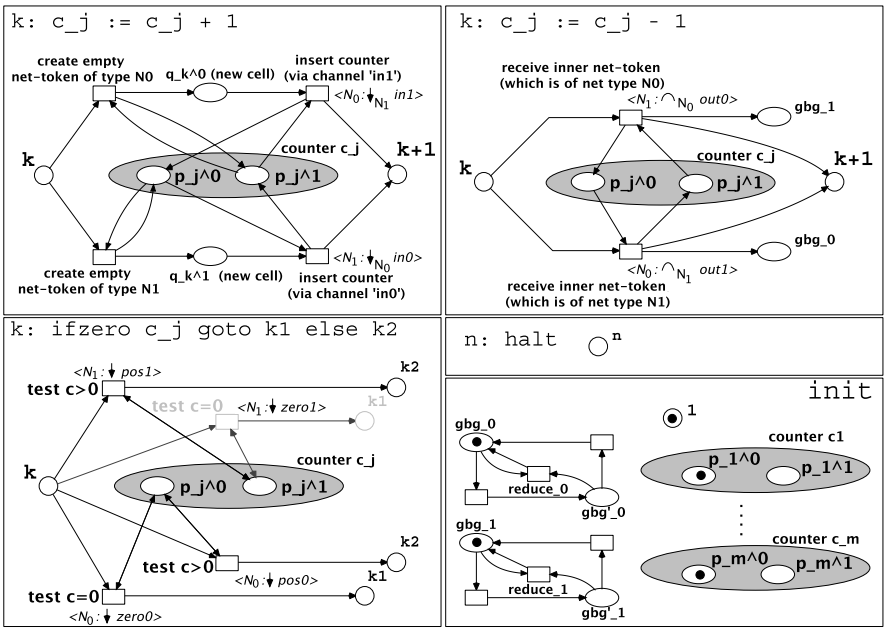


Fig. 8. Net fragments for the simulation of counter programs (from Theorem 1).

Theorem 1. *Object net systems can directly simulate counter programs and thus the reachability problem is undecidable for them.*

Proof sketch. In counter programs one has a fixed number of counters, an increase and a decrease operation and an operation that tests if a certain counter is zero and jumps accordingly.

The counters are encoded by the nesting depth of the two object nets depicted in Figure 7. For a counter c_j two places p_j^0 and p_j^1 will exist in the system net, where p_j^0 is typed with N_0 and p_j^1 with N_1. Initially each place p_j^0 will hold a object net of type N_0 whose place z_0 will be marked by a black token. If the counter is increased a net token of type N_1 is created and the aforementioned net token will be put *into* it on place s_0. The net-tokens are then either packed into each other or unpacked from each other depending on the increase or decrease operation used in the counter program.

Figure 8 shows the net fragments for each of the possible counter program commands. Note how in the increase operation either a net token of type N_0 or of type N_1 is created, depending on the most outer net token currently encoding the state of the counter. The current net-token is then put into the new one by use of the channel in_0 or in_1 again depending on the current state of the counter. If in the current state of the counter the place p_j^0 is marked, then the channel in_0 is used by synchronizing with the identically named channel in the

net of type N_1 residing on q_k^1. The net of type N_0 is taken from p_j^0 transported down via the channel in_0 to the place s_0 in the net of type N_1. This very net then ends up at place p_j^1 representing the new state of the counter. Note how the channel properties of the channel in_0 match. Unpacking for the decrease operation is encoded similarly. The zero test can be easily encoded by trying to synchronise with the transition in N_0 which uses channel $zero_0$. Only the net on the lowest level has the place z_0 marked. Garbage collection (on the lower right of Figure 8) is only needed to make final markings unique and is not discussed here.

Details on the construction can be found in [15]. ◇

Safeness for Object Net Systems. Introducing safeness for ONS and thus restricting the state space to a finite size is part of current work and we only want to touch the topic and not go into details.

In general it is a good idea to restrict the size of the state space by introducing safeness for a given Petri net formalism. Unfortunately due to the nesting depth the state space might non the less become very big for ONS. It seems that the reachability problem is far beyond PSPACE (for 1-safe p/t nets and also for safe EOS, a nets-within-nets formalism with only two levels, reachability is PSPACE-complete [5], [16]) and indeed seems to be EXPTIME-complete.

Definition 3 (Safeness). *Let OS be an* ONS *with initial marking μ_0. OS is* safe *iff $|RS_{OS}(\mu_0)| < \infty$, i.e. if the set of reachable markings of OS is finite.*

OS is strongly safe *iff OS is safe and each net-token is* 1*-safe, that is, if $|RS_{OS}(\mu_0)| < \infty$ and $\forall \mu \in RS(\mu_0)\ \forall \mu' \bigwedge^* \mu\ \forall p \in \mathcal{P} : \Pi^1(\mu')(p) \leq 1$*

Conjecture 1. The Reachability problem for safe ONS is EXPTIME-complete.

Additionally forbidding the creation of net-tokens on the other hand gives us the opportunity to solve the reachability problem in PSPACE again. This restriction is indeed not as severe as it might seem at first glance, because it simply does not allow the creation or destruction of net-tokens which - if net-tokens are interpreted as agents - might not be so undesirable at all.

Definition 4. *Let OS be an* ONS *and $\mu = \sum_{k=1}^{n} p_k[M_k]$ be a marking of OS. With $\Pi_N^3(\mu)$ we denote the* number *of net-tokens of type N present in μ, i.e.*

$$\Pi_N^3(\sum_{k=1}^{n} p_k[M_k]) = \sum_{k=1}^{n} \mathbf{1}_N(p_k) + \Pi_N^3(M_k).$$

Note that $\Pi_N^3(\mu)$ is calculated recursively.

Theorem 2. *Let OS be a strongly safe* ONS *in which no object nets are created nor destroyed, i.e. if $\mu, \mu' \in RS_{OS}(\mu_0)$ and μ' is an immediate successor of μ, then $\Pi_N^3(\mu) = \Pi_N^3(\mu')$ for all $N \in \mathcal{N}$.*[5] *Then the reachability problem is solvable in* PSPACE.

[5] The firing rule ensures that the object nets are actually the "same nets.

Proof sketch. Assume that k net-tokens are present in OS. Furthermore for $N \in \widehat{\mathcal{N}}$ let P_N be the set of places of N. Let $n = max\{P_N \mid N \in \widehat{\mathcal{N}}\}$ be the maximal number of places of the involved nets.

We give an (rough) upper bound for the number of reachable markings. Assume that all net-tokens reside on one system net place $\widehat{p}$. Ignoring the nesting and in particular the structure of nested tokens and thus only taking into account if a place of a net-token is marked or not, we have an upper bound of $(2^n)^k = 2^{n \cdot k}$ different markings, because each net-token is 1-safe and thus has at most 2^n different markings and because we have k net-tokens.

To give a bound for the number of different nestings, we use Cayley's formula according to which the number of different trees on n nodes is n^{n-2} [4]. Note that the nesting of the k net-tokens can be represented by forests, i.e. by a set of trees. The root of each tree represents a net-token residing on $\widehat{p}$. The children of a node v of the tree represent net-tokens residing in the net-token represented by v.

At most we have k trees and each of the k net-tokens may be part of one of those trees, so we have at most k^k different trees. In each of these possibilities we have at most k trees and each tree has at most k nodes, so we have an upper bound of $k^k \cdot k \cdot k^{k-2} < k^{2k}$ for the number of forests on k nodes (where the last factor comes from Cayley's formula).[6]

Taking the number $m := |\widehat{P}|$ of system net places into account we end up with at most $(k^{2k} \cdot 2^{nk})^m \leq (k^{2k} \cdot 2^{nk})^n = k^{2kn} \cdot 2^{nkn} = 2^{\log k \cdot 2kn} \cdot 2^{nkn} < 2^{2kkn+nkn}$

Now note that $2kkn + nkn$ is a polynomial in the input length and thus the technique Savitch used to prove that PSPACE and NPSPACE are equal (cf. [21]) is applicable. Since we can furthermore test in polynomial space if a marking is reachable from another marking and also if a marking is identical to another all necessary operations are possible in polynomial space in the input length, and thus the reachability problem is solvable in polynomial space.[7] ◇

The result above can be easily complemented by a proof of PSPACE-hardness. Indeed the proof in [7] that the reachability problem is PSPACE-hard for ppGSMs can be carried over one-to-one to the setting above.

3 A Mobility Logic for Object Nets

In common logics used in formal verification like CTL and LTL statements about time are possible, e.g. it is possible to ask if a certain state is *ever* reached or if all states reached (in time) have a certain property. In the context of modelling formalisms which allow to model the local distribution of certain objects a logic which also takes locality into account is highly useful. One then might ask questions like e.g. if a certain object will be at a certain position at a certain point in time, or if a certain object will at least be somewhere.

[6] This bound is only a rough approximation, but it suffices here.

[7] For a more detailed discussion of this technique we direct the reader to [7], where we also used it to prove that polynomial space suffices to decide reachability for ppGSMs.

A prominent example for this is the Ambient Calculus and the Ambient Logic associated with it [2], [1], by which our work is deeply inspired. The ambient calculus can be used to describe processes which do not only evolve in time, but also in space. The ambient logic can then be used to express properties of such processes taking into account both, time *and* space.

For the object net formalism presented above a similar logic is desirable. In the example above one could then for example ask the question if it is possible for an agent to enter a certain vehicle. In the following we devise a *Mobility Logic for Object Net Systems* in which satisfaction of formulas will be defined with regard to a given marking of a given object net system, i.e. $OS, \mu \models \mathcal{F}$ holds that the marking μ of the object net system OS satisfies the closed formula $\mathcal{F}$. We usually omit the ONS OS.

The satisfaction relation $\equiv$ is based on the *structural congruence relation*. Intuitively, this relation equalizes markings up to 'commutativity' and 'associativity' of submarkings.

$$\begin{array}{l} \mu \equiv \mu \\ \mu \equiv \mu' \Rightarrow \mu' \equiv \mu \\ \mu \equiv \mu', \mu' \equiv \mu'' \Rightarrow \mu \equiv \mu'' \end{array}$$

$$\begin{array}{l} \mu + \mu' \equiv \mu' + \mu \\ (\mu + \mu') + \mu'' \equiv \mu + (\mu' + \mu'') \\ \mu + \mathbf{0} \equiv \mu \end{array}$$

Formulas are defined inductively by the following grammar:

$$\begin{array}{rl} \phi := & \mathbf{T} \mid \neg\phi \mid (\phi \vee \phi) \mid \\ & \mathbf{0} \mid \ p[\phi] \mid (\phi + \phi) \mid \\ & \Diamond\phi \mid \ \boxdot \phi \end{array}$$

To define the semantic, let OS be an ONS and $\mathcal{M}$ be the set of markings of OS. The truth of a formula ϕ as defined above is then given by the recursively defined relation $\models$ with regard to OS. Note that we used $\mu \bigtriangledown \mu'$ to indicate that the submarking μ' is contained in the marking μ within exactly one level of nesting and that $\mu \bigtriangledown^* \mu'$ means that μ contains μ' at some nesting level.

$$\begin{array}{lll} \forall \mu \in \mathcal{M}\ \mu \models T & & \\ \forall \mu \in \mathcal{M}\ \mu \models \neg\phi & \text{iff} & \mu \not\models \phi \\ \forall \mu \in \mathcal{M}\ \mu \models (\phi_1 \vee \phi_2) & \text{iff} & \mu \models \phi_1 \text{ or } \mu \models \phi_2 \\ \forall \mu \in \mathcal{M}\ \mu \models \mathbf{0} & \text{iff} & \mu \equiv \mathbf{0} \\ \forall \mu \in \mathcal{M}\ \mu \models p[\phi] & \text{iff} & \exists \mu' \in \mathcal{M}.\mu \equiv p[\mu'] \wedge \mu' \models \phi \\ \forall \mu \in \mathcal{M}\ \mu \models (\phi_1 + \phi_2) & \text{iff} & \exists \mu', \mu'' \in \mathcal{M}.\mu \equiv \mu' + \mu'' \wedge \\ & & \quad \mu' \models \phi_1 \wedge \mu'' \models \phi_2 \\ \forall \mu \in \mathcal{M}\ \mu \models \Diamond\phi & \text{iff} & \exists \mu' \in \mathcal{M}.\mu \xrightarrow{*} \mu' \wedge \mu' \models \phi \\ \forall \mu \in \mathcal{M}\ \mu \models \boxdot\phi & \text{iff} & \exists \mu' \in \mathcal{M}.\mu \bigtriangledown^* \mu' \wedge \mu' \models \phi \end{array}$$

Example 1. We give a few examples for formulas of the mobility logic:

- $\mu \models (p[T] + T)$ is true, if the place p is marked in μ at nesting depth 0, that is μ is congruent to $p[\mu'] + \mu''$, where μ' and μ'' are submarkings.
- $\mu \models \Diamond(p_1[\mathbf{0}] + p_2[p[\mathbf{0}]])$ is true, if the place p_1 is marked in μ with an empty net-token (which might also symbolize a black token) and p_2 is marked with an object net whose place p is marked by an empty net-token (or a black token). In this way the standard reachability problem can be formulated.
- $\mu \models \boxdot p[T]$ is true if in the marking μ a object net N resides (in some nesting depth) whose place p is marked (in an arbitrary way). Note that no other place of N might be marked. To allow this one would use the formula $\boxdot(p[T] + T)$.
- $\mu \models \Diamond \boxdot p[T]$ is true if from μ a marking μ' is reachable that satisfies $\boxdot p[T]$ (see above).

Model Checking the Mobility Logic against ONS. Given an ONS OS, a marking μ of OS, and a formula ϕ of the mobility logic, we want to decide if $OS, \mu \models \phi$ holds, i.e. if ϕ is satisfied in the marking μ of OS.

Since we can easily express reachability with the operator $\Diamond$ in the logic, the problem is undecidable for the general ONS-formalism due to Theorem 1 above.

For the restricted ONS-formalisms which enjoy a finite state space, i.e. for safe ONS, strongly safe ONS, and for strongly safe ONS, with a constant number of net-tokens, the question is currently open. We suspect that in the last case PSPACE again suffices and that we need exponential time or exponential space in the first two cases.

Note that to decide if $\mu \models \boxdot p[T] + T$ holds, it is sufficient to scan over μ in linear time and test if p is present somewhere - at least if μ is given as a string as in our examples. This test should thus be easy to do in a subroutine in a PSPACE-algorithm.

The nesting of operators on the other hand, might complicate things since to test $\mu \models \Diamond\boxdot\Diamond\phi$ the formula $\Diamond\phi$ might be true for an object net somewhere which is reached sometime, but only if this object net is then treated in isolation (and not taking the other nets into account). So we might have to start subroutines with different object net systems to decide if certain sub-formulas hold or not.

4 Conclusion and Outlook

The formalisms introduced in this paper, the considered problems, and the results and conjectures so far are summarized in Table 1 below. The entries with a question mark are conjectures.

We have introduced a simplified version of the object net formalism from [15], which still allows the transportation of net-tokens in the vertical dimension, but which has a much easier firing rule, in particular restricting the transitions participating in the firing to at most two levels of nesting.

Table 1. The results and conjectures so far.

	Reachability	Model Checking Mob. Log.
ONS	undecidable	undecidable
safe ONS	EXPTIME-complete ?	EXPTIME-complete ?
strongly safe ONS	EXPTIME-complete ?	EXPTIME-complete ?
strongly safe ONS with a constant number of net-tokens	PSPACE-complete	PSPACE-complete ?

We have shown that the formalism remains Turing-complete and have introduced several restrictions of the formalism to a finite state space: safe ONS, strongly safe ONS, and strongly safe ONS with a constant number of net-tokens. For the last formalism we have shown that the reachability problem is solvable in polynomial space, but all these formalisms deserve a more thorough investigation in the future.

We have then introduced a mobility logic for object net systems which allows to reason about the nesting and thus about the location of net-tokens. In future work we want to focus on the model checking problem for this logic and variants of the object net formalism. We also want to investigate variants of the logic where reasonable. For example it might be interesting to restrict the allowed nesting of operators to prevent e.g. formulas of the form $\Diamond \boxdot \Diamond \phi$.

References

1. Cardelli, L., Gordon, A.D.: Anytime, anywhere. modal logics for mobile ambients. In: Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 365–377. ACM Press (2000)
2. Cardelli, L., Gordon, A.D.: Mobile ambients. Theoretical Computer Science 240, 177–213 (2000)
3. Castagna, G., Vitek, J., Nardelli, F.Z.: The seal calculus. Information and Computation 201, 1–54 (2005)
4. Cayley, A.: A theorem on trees. Quarterly Journal of Pure and Applied Mathematics 23, 376–378 (1889)
5. Esparza, J.: Decidability and complexity of petri net problems – an introduction. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models, Advances in Petri Nets. Lecture Notes in Computer Science, vol. 1491, pp. 374–428. Springer-Verlag (1998)
6. Haddad, S., Poitrenaud, D.: Theoretical aspects of recursive Petri nets. In: Donatelli, S., Kleijn, J. (eds.) Application and Theory of Petri Nets. Lecture Notes in Computer Science, vol. 1639, pp. 228–247. Springer-Verlag (1999)
7. Heitmann, F., Köhler-Bußmeier, M.: P- and t-systems in the nets-within-nets-formalism. In: Pomello, L., Haddad, S. (eds.) To Appear in 33rd International Conference on Application and Theory of Petri Nets and Concurrency. Lecture Notes in Computer Science, Springer-Verlag (2012)
8. Hiraishi, K.: PN^2: An elementary model for design and analysis of multi-agent systems. In: Arbab, F., Talcott, C.L. (eds.) Coordination Models and Languages, COORDINATION 2002. Lecture Notes in Computer Science, vol. 2315, pp. 220–235. Springer-Verlag (2002)

9. Hoffmann, K., Ehrig, H., Mossakowski, T.: High-level nets with nets and rules as tokens. In: Application and Theory of Petri Nets and Other Models of Concurrency. Lecture Notes in Computer Science, vol. 3536, pp. 268 – 288. Springer-Verlag (2005)
10. Köhler, M., Moldt, D., Rölke, H.: Modeling the behaviour of Petri net agents. In: Colom, J.M., Koutny, M. (eds.) Application and Theory of Petri Nets. Lecture Notes in Computer Science, vol. 2075, pp. 224–241. Springer-Verlag (2001)
11. Köhler, M., Moldt, D., Rölke, H.: Modelling mobility and mobile agents using nets within nets. In: v. d. Aalst, W., Best, E. (eds.) Application and Theory of Petri Nets. Lecture Notes in Computer Science, vol. 2679, pp. 121–140. Springer-Verlag (2003)
12. Köhler, M., Rölke, H.: Concurrency for mobile object-net systems. Fundamenta Informaticae 54(2-3) (2003)
13. Köhler, M., Rölke, H.: Properties of Object Petri Nets. In: Cortadella, J., Reisig, W. (eds.) Application and Theory of Petri Nets. Lecture Notes in Computer Science, vol. 3099, pp. 278–297. Springer-Verlag (2004)
14. Köhler-Bußmeier, M.: A survey of elementary object systems: Decidability results. Report of the Department of Informatics, Universität Hamburg (2011)
15. Köhler-Bußmeier, M., Heitmann, F.: On the expressiveness of communication channels for object nets. Fundamenta Informaticae 93(1-3), 205–219 (2009)
16. Köhler-Bußmeier, M., Heitmann, F.: Safeness for object nets. Fundamenta Informaticae 101(1-2), 29–43 (2010)
17. Lakos, C.: A Petri net view of mobility. In: Formal Techniques for Networked and Distributed Systems (FORTE 2005). Lecture Notes in Computer Science, vol. 3731, pp. 174–188. Springer-Verlag (2005)
18. Lomazova, I.A.: Nested Petri nets – a formalism for specification of multi-agent distributed systems. Fundamenta Informaticae 43(1-4), 195–214 (2000)
19. Lomazova, I.A., van Hee, K.M., Oanea, O., Serebrenik, A., Sidorova, N., Voorhoeve, M.: Nested nets for adaptive systems. In: Application and Theory of Petri Nets and Other Models of Concurrency. pp. 241–260. Lecture Notes in Computer Science, Springer-Verlag (2006)
20. Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets I: Basic Models, Lecture Notes in Computer Science, vol. 1491. Springer-Verlag (1998)
21. Savitch, W.: Relationship between nondeterministic and deterministic tape complexities. J. on Computer and System Sciences 4, 177–192 (1970)
22. Valk, R.: Modelling concurrency by task/flow EN systems. In: 3rd Workshop on Concurrency and Compositionality. No. 191 in GMD-Studien, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Bonn (1991)
23. Valk, R.: Object Petri nets: Using the nets-within-nets paradigm. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Advanced Course on Petri Nets 2003. Lecture Notes in Computer Science, vol. 3098, pp. 819–848. Springer-Verlag (2003)

BDD-based Bounded Model Checking for LTLK over Two Variants of Interpreted Systems*

Artur Męski[1,2], Wojciech Penczek[1,3], and Maciej Szreter[1]

[1] Institute of Computer Science, PAS, Ordona 21, 01-237 Warsaw, Poland
{meski,penczek,mszreter}@ipipan.waw.pl
[2] University of Łódź FMCS, Banacha 22, 90-238 Łódź, Poland
[3] University of Natural Sciences and Humanities, Institute of Informatics, 3 Maja 54, 08-110 Siedlce, Poland

Abstract We present a novel approach to verification of multi-agent systems by bounded model checking for Linear Time Temporal Logic extended with the epistemic component (LTLK). The systems are modelled by two variants of interpreted systems: standard and interleaved ones. Our method is based on binary decision diagrams (BDD). We describe the algorithm and provide its experimental evaluation together with the comparison with another tool. This allows to draw some conclusions which semantics is preferable for bounded model checking LTLK properties of multi-agent systems.

1 Introduction

It is often crucial to ensure that multi-agent systems (MAS) conform to their specifications and exhibit some desired behaviour. This can be checked in a fully automatic manner using model checking [5], which is one of the rapidly developing verification techniques. Model checking has been studied by various researchers in the context of MAS and different modal logics for specifying MAS properties [2,7,8,13,15,18,22,23].

When the verification is performed by searching directly through the state space of the MAS, its size is likely to grow exponentially with the number of agents, which is known as the *state-space explosion* problem. Therefore, several approaches alleviating this problem have been proposed. One of them is *bounded model checking* (BMC) [1], in which only a portion of the original model truncated up to some specific depth is considered. This approach can be combined either with a translation of the verification problem to the propositional satisfiability problem (SAT) [20,13] or with symbolic techniques based on *binary decision diagrams* (BDDs) [11].

In this paper we present a novel approach to verification of MAS by BDD-based bounded model checking for Linear Time Temporal Logic extended with

* Partly supported by National Science Centre under the grant No. 2011/01/B/ST6/05317 and 2011/01/B/ST6/01477.

the epistemic component (LTLK, also called CKL$_n$ [8]). The systems are modeled by two variants of Interpreted Systems: standard (IS) [6] and interleaved ones (IIS) [14]. IIS restrict IS by enforcing asynchronous semantics. This does not reduce the expressive power of IS, but modifies this popular modelling approach by bringing the semantics known from verification of concurrent systems like networks of automata or variants of Petri nets. Our paper shows that the modelling approach has a very strong impact on the efficiency of verification. The experimental results exhibit that the IIS-based approach can greatly improve the practical applicability of the bounded model checking method for LTLK.

There has been already some intensive research on BMC for MAS, but mostly for the properties expressible in CTLK, based either on SAT [20,10] or on BDDs [11]. A SAT-based verification method for the LTLK properties of MAS, modeled by IIS, was put forward in [21]. Our technical report [16] presents a BDD-based approach to verification of LTLK for IIS, while the SAT- and BDD-based approaches for IIS are compared in [17].

The rest of the paper is organised as follows. Section 2 provides the basic definitions and notations for LTLK and IS. Our BDD-based BMC method is described in Section 3. The last two sections contain the discussion of an experimental evaluation of the approach and the final remarks.

2 Preliminaries

In this section we introduce the basic definitions used in the paper. In particular, we define the semantics of interpreted systems, as well as the syntax and the semantics of LTLK.

2.1 Formalisms for Modelling Multi-Agent Systems

Interpreted Systems The semantics of *interpreted systems* provides a setting to reason about MAS by means of specifications based on knowledge and linear or branching time. We report here the basic setting as popularised in [6]. We begin by assuming a MAS to be composed of n agents[1] $\mathcal{A}$. We associate a set of *possible local states* L_i and *actions* Act_i to each agent $i \in \mathcal{A}$. We assume that the special action ϵ_i, called "null", or "silent" action of agent i belongs to Act_i; as it will be clear below the local state of agent i remains the same if the null action is performed. Also note we do not assume that the sets of actions of the agents are disjoint. We call $Act = \prod_{i \in \mathrm{A}} Act_i$ the set of all possible *joint actions*, i.e. tuples of local actions executed by agents. We consider a *local protocol* modelling the program the agent is executing. Formally, for any agent i, the actions of the agents are selected according to a *local protocol* $P_i : L_i \to 2^{Act_i}$. For each agent i, we define a relation $t_i \subseteq L_i \times Act \times L_i$, where $(l, (a_1, \ldots, a_n), l) \in t_i$ for each $l \in L_i$ if $a_i = \epsilon_i$. A *global state* $g = (g_1, \ldots, g_n)$ is a tuple of local states for

[1] Note in the present study we do not consider the environment component. This may be added with no technical difficulty at the price of heavier notation.

all the agents corresponding to an instantaneous snapshot of the system at a given time. Given a global state $g = (g_1, \ldots, g_n)$ we denote by $l_i(g)$ the local component g_i of agent $i \in \mathcal{A}$ in g.

For each agent $i \in \mathcal{A}$, $\sim_i \subseteq G \times G$ is an *epistemic indistinguishability* relation over global states defined by $g \sim_i h$ if $l_i(g) = l_i(h)$. Further, let $\Gamma \subseteq \mathcal{A}$. The union of Γ's accessibility relations is defined as $\sim_\Gamma^E = \bigcup_{i \in \Gamma} \sim_i$. By $\sim_\Gamma^C$ we denote the transitive closure of $\sim_\Gamma^E$, whereas $\sim_\Gamma^D = \bigcap_{i \in \Gamma} \sim_i$.

A *global evolution* $\mathcal{T} \subseteq G \times Act \times G$ is defined as follows: $(g, a, h) \in \mathcal{T}$ iff there exists an action $a = (a_1, \ldots, a_n) \in Act$ such that for all $i \in \mathcal{A}$ we have $a_i \in P_i(l_i(g))$ and $(l_i(g), a, l_i(h)) \in t_i$. For $g, h \in G$ and $a \in Act$ s.t. $(g, a, h) \in \mathcal{T}$ we write $g \xrightarrow{a} h$. We assume that the global evolution relation $\mathcal{T}$ is total, i.e., for each $g \in G$ there exists $a \in Act$ and $h \in G$ such that $g \xrightarrow{a} h$.

An infinite sequence of global states and actions $\rho = g_0 a_0 g_1 a_1 g_2 \ldots$ is called a *path* originating from g_0 if there is a sequence of transitions from g_0 onwards, i.e., $g_i \xrightarrow{a_i} g_{i+1}$ for every $i \geq 0$. Any finite prefix of a path is called a *run*. By $length(\rho)$ we mean the number of the states of ρ if ρ is a run, and ω if ρ is a path. In order to limit the indices range of ρ which can be a path or run, we define the relation $\trianglelefteq_\rho$. Let $\trianglelefteq_\rho \stackrel{def}{=} <$ if ρ is a path, and $\trianglelefteq_\rho \stackrel{def}{=} \leq$ if ρ is a run. A state g is said to be *reachable* from g_0 if there is a path or a run $\rho = g_0 a_0 g_1 a_1 g_2 \ldots$ such that $g = g_i$ for some $i \geq 0$. The set of all the paths and runs originating from g is denoted by $\Pi(g)$. The set of all the paths originating from g is denoted by $\Pi^\omega(g)$.

Definition 1 (Interpreted Systems). *Given a set of propositions $\mathcal{PV}$ such that $\{true, false\} \subseteq \mathcal{PV}$, an interpreted system (IS), also referred to as a model, is a tuple $M = (G, \iota, \Pi, \{\sim_i\}_{i \in \mathcal{A}}, \mathcal{V})$, where G is a set of global states, $\iota \in G$ is an initial (global) state such that each state in G is reachable from ι, $\Pi = \bigcup_{g \in G} \Pi(g)$ is the set of all the interleaved paths and runs originating from all the states in G, and $V : G \to 2^{\mathcal{PV}}$ is a valuation function.*

By Π^ω we denote the set of all the paths of Π.

Interleaved Interpreted Systems We define a restriction of interpreted systems, called *interleaved interpreted systems* in which global evolution function is restricted, so that every agent either executes a shared action or the null action.

We assume that $\epsilon_i \in P_i(l)$, for any $l \in L_i$, i.e., we insist on the null action to be enabled at every local state. For each action $a \in \bigcup_{i \in \mathcal{A}} Act_i$ by $Agent(a) \subseteq \mathcal{A}$ we mean all the agents i such that $a \in Act_i$, i.e., the set of the agents potentially able to perform a. Then, the global evolution relation $\mathcal{T}$ is defined as before, but it is restricted by the following condition: if $(g, a, h) \in \mathcal{T}$ then there exists a joint action $a = (a_1, \ldots, a_n) \in Act$, and an action $\alpha \in \bigcup_{i \in \mathcal{A}} Act_i \setminus \{\epsilon_1, \ldots, \epsilon_n\}$ such that: $a_i = \alpha$ for all $i \in Agent(\alpha)$, and $a_i = \epsilon_i$ for all $i \in \mathcal{A} \setminus Agent(\alpha)$. Similar to blocking synchronisation in automata, the above insists on all the agents performing the same non-epsilon action in a global evolution; additionally, note that if an agent has the action being performed in its repertoire it must be performed for the global evolution to be allowed. This assumes local protocols

are defined in such a way to permit this; if a local protocol does not allow this, the local action cannot be performed and therefore the global evolution does not comply with the above definition of interleaving.

2.2 Syntax and Semantics of LTLK

Combinations of linear time with knowledge have long been used in the analysis of temporal epistemic properties of systems [6]. We now recall the basic definitions here and adapt them to our purposes when needed.

Definition 2 (Syntax). *Let $\mathcal{PV}$ be a set of atomic propositions to be interpreted over the global states of a system, $p \in \mathcal{PV}$, $q \in \mathcal{A}$, and $\Gamma \subseteq \mathcal{A}$. Then, the syntax of* LTLK *is defined by the following BNF grammar:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi \mid \varphi \mathrm{R} \varphi \mid \mathrm{K}_q\varphi \mid \overline{\mathrm{K}}_q\varphi \mid \mathrm{E}_\Gamma\varphi \mid \overline{\mathrm{E}}_\Gamma\varphi \mid \mathrm{D}_\Gamma\varphi \mid \overline{\mathrm{D}}_\Gamma\varphi \mid \mathrm{C}_\Gamma\varphi \mid \overline{\mathrm{C}}_\Gamma\varphi.$$

The temporal operators U and R are named as usual *until* and *release* respectively, X is the next step operator. The epistemic operators K_q, $\mathrm{D}_\Gamma, \mathrm{E}_\Gamma$, and C_Γ represent, respectively, knowledge of agent q, distributed knowledge in the group Γ, "everyone in Γ knows", and common knowledge among agents in Γ, whereas $\overline{\mathrm{K}}_q$, $\overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma$, and $\overline{\mathrm{C}}_\Gamma$ are the corresponding dual ones.

Typically, the semantics of LTLK is defined over paths of a model M only, whereas our semantics exploits paths and runs. This semantics can be conveniently applied also to submodels (to be defined later) in order to verify efficiently the existential fragment of LTLK over paths and runs.

Definition 3 (Semantics). *Given a model $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$, where $\mathcal{V}(s)$ is the set of propositions that hold at s, let $\rho(i)$ denote the i-th state of a path or run $\rho \in \Pi$, and $\rho[i]$ denote the path or run ρ with a designated formula evaluation position i, where $i \trianglelefteq_\rho length(\rho)$. Note that $\rho[0] = \rho$. The formal semantics of* LTLK *is defined recursively as follows:*

- $M, \rho[i] \models p$ *iff* $p \in \mathcal{V}(\rho(i))$,
- $M, \rho[i] \models \neg\varphi$ *iff* $M, \rho[i] \not\models \varphi$,
- $M, \rho[i] \models \varphi_1 \wedge \varphi_2$ *iff* $M, \rho[i] \models \varphi_1$ *and* $M, \rho[i] \models \varphi_2$,
- $M, \rho[i] \models \varphi_1 \vee \varphi_2$ *iff* $M, \rho[i] \models \varphi_1$ *or* $M, \rho[i] \models \varphi_2$,
- $M, \rho[i] \models \mathrm{X}\varphi$ *iff* $length(\rho) > i$ *and* $M, \rho[i+1] \models \varphi$;
- $M, \rho[i] \models \varphi_1 \mathrm{U} \varphi_2$ *iff* $(\exists k \geq i)[M, \rho[k] \models \varphi_2$ *and* $(\forall i \leq j < k)\ M, \rho[j] \models \varphi_1]$,
- $M, \rho[i] \models \varphi_1 \mathrm{R} \varphi_2$ *iff* $[(\rho \in \Pi^\omega(\iota)$ *and* $(\forall k \geq i)\ M, \rho[k] \models \varphi_2]$ *or* $(\exists k \geq i)[M, \rho[k] \models \varphi_1$ *and* $(\forall i \leq j \leq k)\ M, \rho[j] \models \varphi_2]$,
- $M, \rho[i] \models \mathrm{K}_q\varphi$ *iff* $(\forall \rho' \in \Pi^\omega(\iota))(\forall k \geq 0)[\rho'(k) \sim_q \rho(i)$ *implies* $M, \rho'[k] \models \varphi]$,
- $M, \rho[i] \models \overline{\mathrm{K}}_q\varphi$ *iff* $(\exists \rho' \in \Pi(\iota))(\exists k \geq 0)[\rho'(k) \sim_q \rho(i)$ *and* $M, \rho'[k] \models \varphi]$,
- $M, \rho[i] \models \mathrm{Y}_\Gamma\varphi$ *iff* $(\forall \rho' \in \Pi^\omega(\iota))(\forall k \geq 0)[\rho'(k) \sim_\Gamma^{\mathrm{Y}} \rho(i)$ *implies* $M, \rho'[k] \models \varphi]$,
- $M, \rho[i] \models \overline{\mathrm{Y}}_\Gamma\varphi$ *iff* $(\exists \rho' \in \Pi(\iota))(\exists k \geq 0)[\rho'(k) \sim_\Gamma^{\mathrm{Y}} \rho(i)$ *and* $M, \rho'[k] \models \varphi]$, *where* $\mathrm{Y} \in \{\mathrm{D}, \mathrm{E}, \mathrm{C}\}$.

Let $g \in G$ and φ be an LTLK formula. We use the following notations:

– $M, g \models \varphi$ iff $M, \rho[0] \models \varphi$ for all the paths $\rho \in \Pi^{\omega}(g)$;
– $M \models \varphi$ iff $M, \iota \models \varphi$;
– $Props(\varphi)$ is the set of atomic propositions appearing in φ.

LTL is the sublogic of LTLK which consists only of the formulae built without the epistemic operators. ELTLK is the existential fragment of LTLK, defined by the following grammar:

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi \mid \varphi \mathrm{R} \varphi \mid \overline{\mathrm{K}}_q \varphi \mid \overline{\mathrm{E}}_\Gamma \varphi \mid \overline{\mathrm{D}}_\Gamma \varphi \mid \overline{\mathrm{C}}_\Gamma \varphi.$$

Moreover, an ELTLK formula φ holds in the model M, denoted $M \models_\exists \varphi$, iff $M, \rho[0] \models \varphi$ for some path or run $\rho \in \Pi(\iota)$. The intuition behind this definition is that ELTLK is obtained only by restricting the syntax of the epistemic operators while the temporal ones remain the same. We get the existential version of these operators by the change from the universal quantification over the paths ($\models$) to the existential quantification ($\models_\exists$) over the paths and the runs in the definition of the validity in the model M. Notice that this change is only necessary when φ contains a temporal operator, which is not nested in an epistemic operator.

Our semantics meets two important properties. Firstly, for LTL the definition of validity in a model M uses paths only. Secondly, if we replace each Π with Π^{ω}, the semantics does not change as our models have total transition relations (each run is a prefix of some path). The semantics applied to submodels of M does not have the above property, but it preserves ELTLK over M, which is shown in Lemma 1.

3 BDD-based BMC for ELTLK

In this section we show how to perform BMC of ELTLK using BDDs [5] by combining the standard approach for ELTL [4] with the method for the epistemic operators [22] in a similar manner to the solution for CTL* of [5].

Let $\mathcal{PV}$ be a set of propositions. For an ELTLK formula φ we define inductively the *number* $\gamma(\varphi)$ *of nested epistemic operators* in the formula:

– if $\varphi = p$, where $p \in \mathcal{PV}$, then $\gamma(\varphi) = 0$,
– if $\varphi = \odot \varphi'$ and $\odot \in \{\neg, \mathrm{X}\}$, then $\gamma(\varphi) = \gamma(\varphi')$,
– if $\varphi = \varphi' \odot \varphi''$ and $\odot \in \{\wedge, \vee, \mathrm{U}, \mathrm{R}\}$, then $\gamma(\varphi) = \gamma(\varphi') + \gamma(\varphi'')$,
– if $\varphi = \mathrm{Y}\varphi'$ and $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$, then $\gamma(\varphi) = \gamma(\varphi') + 1$.

Let $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$. If $\varphi = \mathrm{Y}\psi$ is an ELTLK formula, by $sub(\varphi)$ we denote the formula ψ nested in the epistemic operator Y. Moreover, for an arbitrary ELTLK formula φ we define inductively the set $\mathcal{Y}(\varphi)$ of its subformulae in the form $\mathrm{Y}\psi$:

– if $\varphi = p$, where $p \in \mathcal{PV}$, then $\mathcal{Y}(\varphi) = \emptyset$,
– if $\varphi = \odot \varphi'$ and $\odot \in \{\neg, \mathrm{X}\}$, then $\mathcal{Y}(\varphi) = \mathcal{Y}(\varphi')$,
– if $\varphi = \varphi' \odot \varphi''$ and $\odot \in \{\wedge, \vee, \mathrm{U}, \mathrm{R}\}$, then $\mathcal{Y}(\varphi) = \mathcal{Y}(\varphi') \cup \mathcal{Y}(\varphi'')$,
– if $\varphi = \mathrm{Y}\varphi'$ and $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$, then $\mathcal{Y}(\varphi) = \mathcal{Y}(\varphi') \cup \{\varphi\}$.

Definition 4. *Let* $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$ *and* $U \subseteq G$ *with* $\iota \in U$*. The* submodel *generated by* U *is a tuple* $M|_U = (U, \iota, \Pi', \{\sim'_q\}_{q\in\mathcal{A}}, \mathcal{V}')$*, where:* $\sim'_q = \sim_q \cap U^2$ *for each* $q \in \mathcal{A}$*,* $\mathcal{V}' = \mathcal{V} \cap U^2$*, and* Π' *is the set of the paths and runs of* M *having all the states in* U*, formally,* $\Pi' = \{\rho \in \Pi \mid (\forall 0 \leq i \trianglelefteq_\rho length(\rho))\ \rho(i) \in U\}$*.*

For ELTLK formulae φ, ψ, and ψ', by $\varphi[\psi \leftarrow \psi']$ we denote the formula φ in which every occurrence of ψ is replaced with ψ'. Let $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$ be a model, then by $\mathcal{V}_M$ we understand the valuation function $\mathcal{V}$ of the model M, and by $G_R \subseteq G$ the set of its reachable states. Moreover, we define $[\![M, \varphi]\!] = \{g \in G_R \mid M, g \models^\exists \varphi\}$.

Reducing ELTLK to ELTL. Given a model $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$, and an ELTLK formula φ, Algorithm 1 is used to compute the set $[\![M, \varphi]\!]$, under the assumption that we have the algorithms for computing this set for each φ being an ELTL formula or in the form $\mathrm{Y}p$, where $p \in \mathcal{PV}$, and $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$ (we use the algorithms from [4] and [22], respectively). In order to obtain this set, we construct a new model M_c together with an ELTL formula φ_c, as described in Algorithm 1, and compute the set $[\![M_c, \varphi_c]\!]$, which is equal to $[\![M, \varphi]\!]$. Initially φ_c equals φ, which is an ELTLK formula, and we process the formula in stages to reduce it to an ELTL formula by replacing with atomic propositions all its subformulae containing epistemic operators. We begin by choosing some epistemic subformula ψ of φ_c, which consists of exactly one epistemic operator, and process it in two stages. First, we modify the valuation function of M_c such that every state initialising some path or run along which $sub(\psi)$ holds is labelled with the new atomic proposition $p_{sub(\psi)}$, and we replace with the variable $p_{sub(\psi)}$ every occurrence of $sub(\psi)$ in ψ. In the second stage, we deal with the epistemic operators having in their scopes atomic propositions only. By modifying the valuation function of M_c we label every state initialising some path or run along which the modified simple epistemic formula ψ holds with a new variable p_ψ. Similarly to the previous stage, we replace every occurrence of ψ in φ_c with p_ψ. In the subsequent iterations, we process every remaining epistemic subformulae of φ_c in the same way until there are no more nested epistemic operators in φ_c, i.e., we obtain an ELTL formula φ_c, and the model M_c with the appropriately modified valuation function. Finally, we compute the set of all reachable states of M_c that initialise at least one path or run along which φ_c holds (line 13). The correctness of the substitution used in Algorithm 1 is stated by the following proposition:

Proposition 1. *Let* $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$ *be a model,* φ *an* ELTLK *formula, and* $\rho \in \Pi$ *some path or run with an evaluation position such that* $m \trianglelefteq_\rho length(\rho)$*. We define* $p \in \mathcal{PV}$ *such that* $M, \rho'[m'] \models p$ *iff* $M, \rho'[m'] \models \varphi$ *for all* $\rho' \in \Pi(\iota)$*, where* $m' \trianglelefteq_{\rho'} length(\rho')$*. Then,* $M, \rho[m] \models \mathrm{Y}\varphi$ *iff* $M, \rho[m] \models \mathrm{Y}p$*, where* $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$*, and* $q \in \mathcal{A}$*,* $\Gamma \subseteq \mathcal{A}$*.*

Proof. Straightforward from the semantics of ELTLK.

Algorithm 1. Computation of $[\![M, \varphi]\!]$

1: $M_c := M$, $\varphi_c := \varphi$
2: **while** $\gamma(\varphi_c) \neq 0$ **do**
3: pick $\psi \in \mathcal{Y}(\varphi_c)$ such that $\gamma(\psi) = 1$
4: **for all** $g \in [\![M_c, sub(\psi)]\!]$ **do**
5: $\mathcal{V}_{M_c}(g) := \mathcal{V}_{M_c}(g) \cup \{p_{sub(\psi)}\}$
6: **end for**
7: $\psi := \psi[sub(\psi) \leftarrow p_{sub(\psi)}]$
8: **for all** $g \in [\![M_c, \psi]\!]$ **do**
9: $\mathcal{V}_{M_c}(g) := \mathcal{V}_{M_c}(g) \cup \{p_\psi\}$
10: **end for**
11: $\varphi_c := \varphi_c[\psi \leftarrow p_\psi]$
12: **end while**
13: **return** $[\![M_c, \varphi_c]\!]$

Algorithm 2. BMC algorithm

1: $Reach := \{\iota\}$,$New := \{\iota\}$
2: **while** $New \neq \emptyset$ **do**
3: $Next := New_{\rightsquigarrow}$
4: **if** $\iota \in [\![M|_{Reach}, \varphi]\!]$ **then**
5: **return** $true$
6: **end if**
7: $New := Next \setminus Reach$
8: $Reach := Reach \cup New$
9: **end while**
10: **return** $false$

BMC Algorithm. To perform bounded model checking of an ELTLK formula, we use Algorithm 2. Given a model M and an ELTLK formula φ, the algorithm checks if there exists a path or run initialised in ι on which φ holds, i.e., if $M, \iota \models^{\exists} \varphi$. For any $X \subseteq G$ by $X_{\rightsquigarrow} \stackrel{def}{=} \{g' \in G \mid (\exists g \in X)(\exists \rho \in \Pi(g))\ g' = \rho(1)\}$ we define the set of the immediate successors of all the states in X. The algorithm starts with the set $Reach$ of reachable states that initially contains only the state ι. With each iteration the verified formula is checked (line 4), and the set $Reach$ is extended with new states (line 8). The algorithm operates on submodels $M|_{Reach}$ generated by the set $Reach$ to check if the initial state ι is in the set of states from which there is a path or run on which φ holds. The loop terminates if there is such a path or run in the obtained submodel, and the algorithm returns $true$ (line 5). The search continues until no new states can be reached from the states in $Reach$. When we obtain the set the of reachable states, and a path or run from the initial state on which φ holds could not be found in any of the obtained submodels, the algorithm terminates returning $false$.

The correctness of the results obtained by the bounded model checking algorithm is formulated by the following lemma:

Lemma 1. *Let $M = (G, \iota, \Pi, \{\sim_q\}_{q \in \mathcal{A}}, \mathcal{V})$ be a model, φ an* ELTLK *formula, and $\rho \in \Pi$ a path or run with an evaluation position m such that $m \trianglelefteq_\rho length(\rho)$. Then, $M, \rho[m] \models \varphi$ iff exists $G' \subseteq G$ such that $\iota \in G'$, and $M|_{G'}, \rho[m] \models \varphi$.*

Proof. "$\Rightarrow$" This way the proof is obvious as we simply take $G' = G$.
"$\Leftarrow$" This way the proof is more involved. It is by induction on the length of a formula φ. The base case is straightforward, as the lemma follows directly for the propositional variables and their negations. Assume, the statement holds for all the proper subformulae of φ. Let $G' \subseteq G$ be a set of states such that $M|_{G'}$ contains ρ, and (*) $M|_{G'}, \rho[i] \models \varphi$, where $i \in \mathbb{N}$.

1. Let $\varphi = \alpha \vee \beta$. By the semantics and the assumption (*), $M|_{G'}, \rho[i] \models \alpha$ or $M|_{G'}, \rho[i] \models \beta$. Using the induction hypothesis and the definition of

submodel (Def. 4), ρ exists also in the model M, and $M, \rho[i] \models \alpha$ or $M, \rho[i] \models \beta$, thus $M, \rho[i] \models \alpha \vee \beta$.

2. Let $\varphi = \alpha \wedge \beta$. By the semantics and the assumption (*), $M|_{G'}, \rho[i] \models \alpha$ and $M|_{G'}, \rho[i] \models \beta$. Using the induction hypothesis and the definition of submodel, ρ exists also in the model M. Therefore, $M, \rho[i] \models \alpha$ and $M, \rho[i] \models \beta$, thus $M, \rho[i] \models \alpha \wedge \beta$.
3. Let $\varphi = \mathrm{X}\alpha$. By the semantics and the assumption (*), $length(\rho) > i$, and $M|_{G'}, \rho[i+1] \models \alpha$. Using the induction hypothesis and the definition of submodel, we get that ρ exists also in M, and $M, \rho[i+1] \models \alpha$, therefore $M, \rho[i] \models \mathrm{X}\alpha$.
4. Let $\varphi = \alpha \mathrm{U} \beta$. By the semantics and the assumption (*), there exists $k \geq i$, such that $M|_{G'}, \rho[k] \models \beta$, and $M|_{G'}, \rho[j] \models \alpha$, for all $i \leq j < k$. Using the induction hypothesis and the definition of submodel, we get that ρ exists also in M. Therefore, from $M, \rho[k] \models \beta$, and $M, \rho[j] \models \alpha$ for all $i \leq j < k$, it follows that $M, \rho[i] \models \alpha \mathrm{U} \beta$.
5. Let $\varphi = \alpha \mathrm{R} \beta$. By the semantics and the assumption (*) we have one or both of the following cases:
   (a) ρ is a path of $M|_{G'}$, and $M|_{G'}, \rho[k] \models \beta$ for all $k \geq i$, then from the definition of submodel, ρ exists also in M, and $\rho \in \Pi^{\omega}$. Using the induction hypothesis, we have that $M, \rho[k] \models \beta$ for all $k \geq i$. Therefore, it follows that $M, \rho[i] \models \alpha \mathrm{R} \beta$.
   (b) There exists $k \geq i$ such that $M|_{G'}, \rho[k] \models \alpha$, and $M|_{G'}, \rho[j] \models \beta$ for all $i \leq j \leq k$. From the definition of submodel, ρ also exists in M, and using the induction hypothesis we get that $M, \rho[k] \models \alpha$, and $M, \rho[j] \models \beta$ for all $i \leq j \leq k$. Thus, $M, \rho[i] \models \alpha \mathrm{R} \beta$.
6. Let $q \in \mathcal{A}$, and $\varphi = \overline{\mathrm{K}}_q \alpha$. By the semantics and the assumption (*), there exists such a path or run ρ' in $M|_{G'}$ that $\rho'(k) \sim_q \rho(i)$ for some $k \geq 0$, and $M|_{G'}, \rho'[k] \models \alpha$. From the definition of submodel, ρ and ρ' also exist in M. Using the induction hypothesis, we get that $M, \rho'[k] \models \alpha$ and $\rho'(k) \sim_q \rho(i)$. Thus, $M, \rho[i] \models \overline{\mathrm{K}}_q \alpha$.
7. Let $\Gamma \subseteq \mathcal{A}$, and $\varphi = \overline{\mathrm{Y}}_{\Gamma} \alpha$, where $\mathrm{Y} \in \{\mathrm{D}, \mathrm{E}, \mathrm{C}\}$. By the semantics and the assumption (*), there exists such a path or run ρ' in $M|_{G'}$ that $\rho'(k) \sim_{\Gamma}^{\mathrm{Y}} \rho(i)$ for some $k \geq 0$, and $M|_{G'}, \rho'[k] \models \alpha$. From the definition of submodel, ρ and ρ' also exist in M. Using the induction hypothesis, we get that $M, \rho'[k] \models \alpha$ and $\rho'(k) \sim_{\Gamma}^{\mathrm{Y}} \rho(i)$. Thus, $M, \rho[i] \models \overline{\mathrm{Y}}_{\Gamma} \alpha$.

Model Checking ELTL. To compute the sets of states corresponding to the ELTL formulae, needed in Algorithm 1, we use the method described in [4] and based on checking the non-emptiness of Büchi automata. Given a model M and an ELTL formula φ, we begin with constructing the tableau for φ (as described in [4]), that is then combined with M to obtain their product, which contains these paths of M where φ potentially holds. Next, the product is verified in terms of the CTL model checking of EG$true$ formula under fairness constraints. Those constraints, corresponding to sets of states, allow to choose only the paths of the model, along which at least one state in each set representing fairness constraints

appears in a cycle. In case of ELTL model checking, fairness guarantees that $\varphi \mathrm{U} \psi$ really holds, i.e., eliminates the paths where φ holds continuously, but ψ never holds. Finally, we choose only these reachable states of the product that belong to some particular set of states computed for the formula. The corresponding states of the verified system that are in this set, comprise the set $[\![M, \varphi]\!]$, i.e., the reachable states where the verified formula holds. As we are unable to include more details (due to the page limit), we refer the reader to [4].

The method described above has some limitations when used for BMC, where it is preferable to detect counterexamples using not only the paths but also the runs of the submodel. As totality of the transition relation of the verified model is assumed, counterexamples are found only along the paths of the model. However, this remains correct even if the final submodel only has the total transition relation: in the worst case the detection of the counterexample is delayed to the last iteration, i.e., when all the reachable states are computed. Nonetheless, this should not keep us from assessing the potential efficiency of our approach.

Model Checking Epistemic Modalities. In order to verify the formulae of the form $\mathrm{Y}p$, where $p \in \mathcal{PV}$, and $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$, we use the algorithms described in [22]. The procedures simply follow from the semantics of ELTLK. The algorithm for $\overline{\mathrm{C}}_\Gamma$ involves a fix point computation, whereas for the remaining operators the algorithms are based on simple non-iterative computations.

4 Experimental Results

In this section we consider three scalable systems which we use to evaluate the efficiency of our BDD-based BMC for LTLK over two variants of Interpreted Systems: IS and IIS. We also compare our results with the ones obtained using MCK. The tool MCK[2] enables fair comparisons for IS semantics, as according to the manual it supports SAT-based BMC for CTL*K. Unfortunately, no theory behind this implementation has ever been published. The paper [10], which describes SAT-based BMC for CTLK, does not discuss how this approach can be extended to CTL*K.

The tests have been performed on a computer fitted with Intel Xeon 2 GHz processor and 4 GB of RAM, running Linux 2.6. Our methods are implemented with reordering, and with the fixed interleaving order of the BDD variables. The reordering is performed by the Rudell's sifting algorithm available in CUDD library, used for manipulating BDDs.

The specifications for the described benchmarks are given in the universal form, for which we verify the corresponding counterexample formula, i.e., the formula which is negated and interpreted existentially. Moreover, for every specification given, there exists a counterexample. With $i(n)$ and $i^{\mathrm{I}}(n)$ we denote the number of iterations needed by our algorithms for IS and IIS, respectively, to find

[2] `http://cgi.cse.unsw.edu.au/~mck/mcks/docDownload/manual`, version 0.5.1 was used as the newer 1.0.0 is provided only for 32-bit machines.

the counterexample, where n is the scaling parameter. The detailed descriptions of our experiments together with the specifications for the systems used, can be found at the web page of VerICS.[3] The memory and the time consumption are shown in the respective figures as the functions of the scaling parameter for each benchmark. Note that the figures are presented in a logarithmic scale.

4.1 Benchmarks

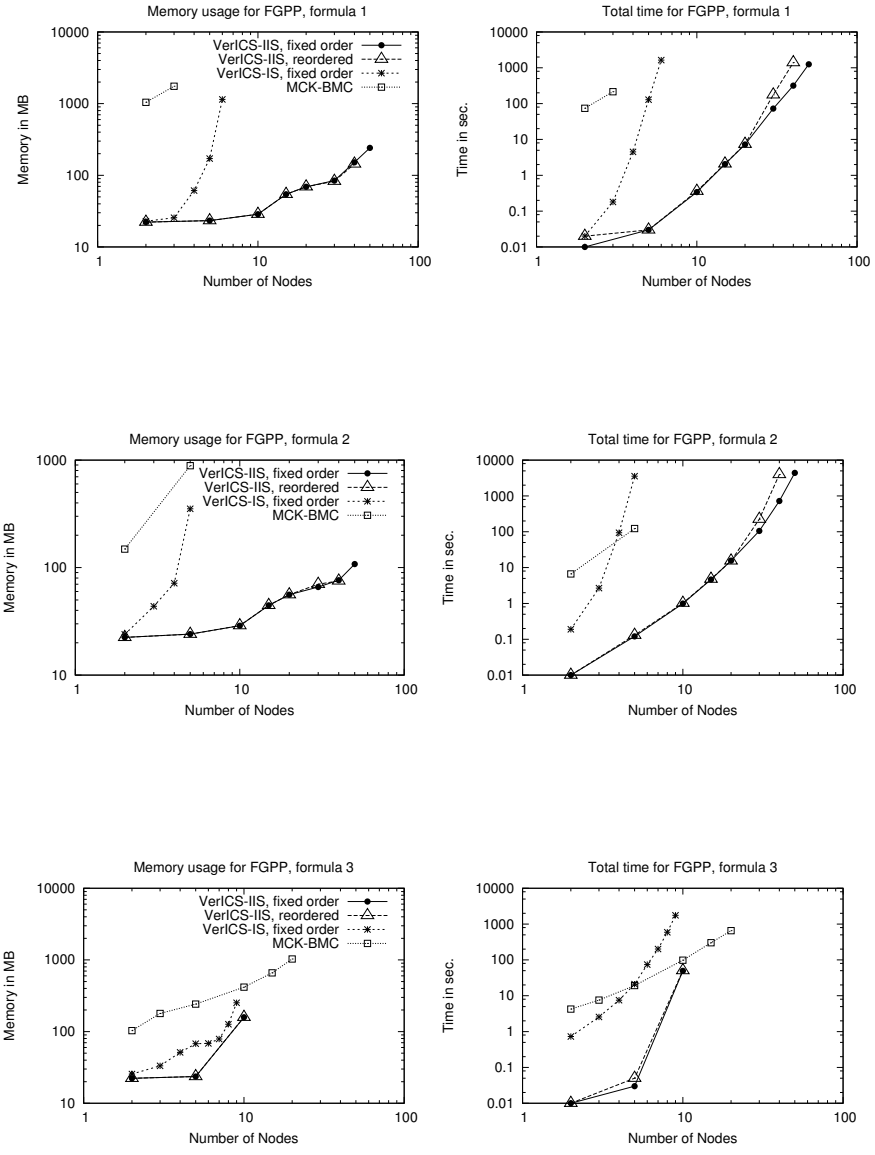
Faulty Generic Pipeline Paradigm (FGPP) (adapted from [19]) consists of Producer, Consumer, and a chain of n intermediate Nodes transmitting data, together with a chain of n Alarms enabled when some error occurs. We consider the following specifications:
$\varphi_1 = \mathrm{G}(ProdSend \rightarrow \mathrm{K}_C\mathrm{K}_P ConsReady))$, $\varphi_2 = \mathrm{G}(Problem_n \rightarrow (\mathrm{F}(Repair_n) \vee \mathrm{G}(Alarm_n Send)))$, $\varphi_3 = \bigwedge_{i=1}^{n} \mathrm{G}(Problem_i \rightarrow (\mathrm{F}(Repair_i) \vee \mathrm{G}(Alarm_i Send)))$, and $\varphi_4 = \bigwedge_{i=1}^{n} \mathrm{G}(\mathrm{K}_P(Problem_i \rightarrow (\mathrm{F}(Repair_i) \vee \mathrm{G}(Alarm_i Send))))$. The formula φ_1 ($i(n) = i^{\mathtt{I}}(n) = 2n + 3$) states that if Producer produces a commodity, then Consumer knows that Producer does not know that Consumer has the commodity. The formula φ_2 ($i(n) = i^{\mathtt{I}}(n) = 2n + 4$) expresses that each time a problem occurs at node n, then either it is repaired or the alarm of node n rings. The formula φ_3 ($i^{\mathtt{I}}(n) = 8$, $i(n) = 2n + 4$) expresses that each time a problem occurs on a node, then either it is repaired or the alarm rings. The formula, φ_4 ($i^{\mathtt{I}}(n) = 5$, $i(n) = 8$) expresses that Producer knows that each time a problem occurs on a node, then either it is repaired or the alarm rings.

A faulty train controller system (FTC) (adapted from [9]) consists of a controller and n trains (for $n \geq 2$), one of which is dysfunctional. We consider the following specifications: $\varphi_1 = \mathrm{G}(InTunnel_1 \rightarrow \mathrm{K}_{Train_1}(\bigwedge_{i=2}^{n} \neg InTunnel_i))$, and $\varphi_2 = \mathrm{G}(\mathrm{K}_{Train_1} \bigwedge_{i=1,j=2,i<j}^{n} \neg(InTunnel_i \wedge InTunnel_j)$. The formula φ_1 ($i^{\mathtt{I}}(n) = 5$, $i(n) = 8$) expresses that whenever a train is in the tunnel, it knows that the other trains are not. The formula φ_2 ($i^{\mathtt{I}}(n) = 5$, $i(n) = 7$) represents that trains are aware of the fact that they have exclusive access to the tunnel.

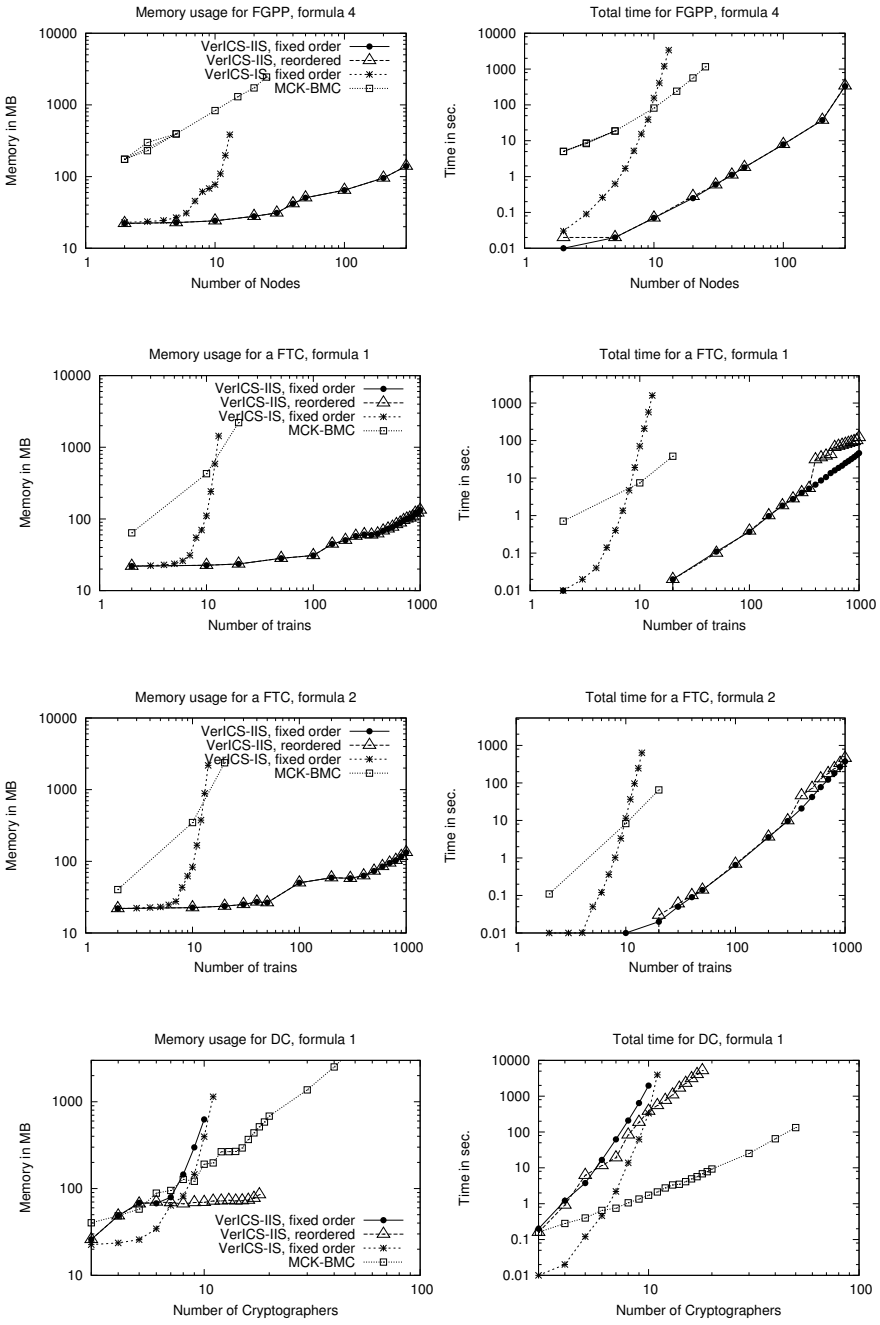
Dining Cryptographers (DC) [3] is a scalable anonymity protocol, which has been formalised and analysed in many works, e.g., [12,15]. We consider the following specifications: $\varphi_1 = \mathrm{G}(odd \wedge \neg paid_1 \rightarrow \bigvee_{i=2}^{n} \mathrm{K}_1(paid_i))$, $\varphi_2 = \mathrm{G}(\neg paid_1 \rightarrow \mathrm{K}_1(\bigvee_{i=2}^{n} paid_i))$, $\varphi_3 = \mathrm{G}(odd \rightarrow \mathrm{C}_{1,\ldots,n}\neg(\bigvee_{i=1}^{n} paid_i))$. The formula φ_1 ($i^{\mathtt{I}}(n) = 4n + 2$, $i(n) = 3$) expresses that always when the number of uttered differences is odd and the first cryptographer has not paid for dinner, then he knows the cryptographer who paid for dinner. The formula φ_2 ($i^{\mathtt{I}}(n) = 2$, $i(n) = 3$) states that it is always true that if the first cryptographer has not paid for dinner, then he knows that some other cryptographer pays. The formula φ_3 ($i^{\mathtt{I}}(n) = 4n + 2$, $i(n) = 3$) states that always when the number of uttered differences is odd, then it is common knowledge of all the cryptographers that none of the cryptographers has paid for dinner.

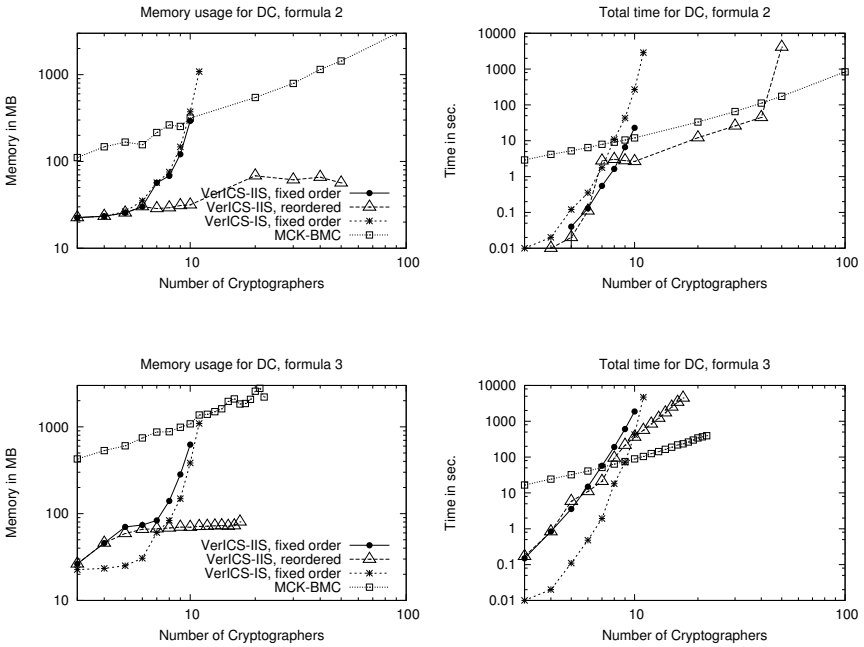
[3] `http://verics.ipipan.waw.pl/r/2is`

4.2 Performance Evaluation



Comparing IS algorithms, in most cases MCK is better than VerICS-IS, but remains close when looking at the orders of magnitude. The reason for better performance of MCK may come from the fact that it is based on the translation


Memory usage for FGPP, formula 4
Memory in MB
Number of Nodes
VerICS-IIS, fixed order
VerICS-IIS, reordered
VerICS-IS, fixed order
MCK-BMC
Total time for FGPP, formula 4
Time in sec.
Number of Nodes
Memory usage for a FTC, formula 1
Memory in MB
Number of trains
Total time for a FTC, formula 1
Time in sec.
Number of trains
Memory usage for a FTC, formula 2
Memory in MB
Number of trains
Total time for a FTC, formula 2
Time in sec.
Number of trains
Memory usage for DC, formula 1
Memory in MB
Number of Cryptographers
Total time for DC, formula 1
Time in sec.
Number of Cryptographers



to SAT, and SAT-based BMC does not need to store the whole examined part of the state space.

For most of the considered benchmarks the VerICS-IIS method is superior to the two IS approaches: MCK and VerICS-IS, sometimes even by several orders of magnitude. This can be observed especially in the case of FTC. However, in the case of FGPP and φ_3 with no epistemic modalities, MCK proved to be more efficient, but for the formula φ_4 containing the K operator, VerICS-IIS was superior. This can be justified by the fact that introducing epistemic modalities partitions the ELTL verification task into several smaller ones.

In the case of IIS, the reordering of the BDD variables does not cause any significant change of the performance in the case of FGPP and FTC, but for DC it reduces the memory consumption. Therefore, for IIS the fixed interleaving order we used can often be considered optimal. The penalty in the verification time to reorder the variables, in favour of reducing memory consumption, is also not significant and can be worth the tradeoff. However, in the case of IS the performance did not change, thus we include only the results for the fixed order of the variables for VerICS-IS.

It is important to note that from our comparison of [17] it follows that in the case of IIS, the general performance of BDD-based approach is superior to the SAT-based one. Therefore, we can conclude now that BMC for LTLK is less efficient for IS when comparing with IIS. This could be explained by the different structure of the state space, which for IS is more dense, i.e., more states

are explored at every iteration of the BMC algorithm. The case of DC shows that this factor can be more important than the lengths of the counterexamples, which can be shorter for IS, or may even be of constant length when scaling the system.

5 Final Remarks

We have proposed, implemented, and experimentally evaluated our BDD-based BMC algorithms for LTLK over two variants of Interpreted Systems: standard and interleaved ones. The experimental results show that the approach based on the interleaved Interpreted Systems can greatly improve the practical applicability of the bounded model checking method. Although, we have tested only properties of LTLK, we can expect to obtain similar results for other specification formalisms. Moreover, contrary to the SAT-based method of MCK and of [21], our BDD-based BMC is complete, i.e., it can also be easily used to verify that existential properties are false in the considered model.

In the future we are going to extend the presented algorithms to handle also the CTL*K properties. Since our implementation is in its preliminary stage, we also need to improve it in many ways, e.g., it should be investigated in the case of the non-interleaving semantics whether a different strategy for finding good BDD variables ordering would improve the results.

Our results are preliminary and the comparison is by no means complete. It ignores the fact that some formulae can give different verification results for each of the considered semantics, e.g., in the presence of the next-state operator X. However, we believe our results can be viewed as a justification and a starting point for further research on the subject.

References

1. A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. In *Highly Dependable Software*, volume 58 of *Advances in Computers*. Academic Press, 2003. Pre-print.
2. R. Bordini, M. Fisher, C. Pardavila, W. Visser, and M. Wooldridge. Model checking multi-agent programs with CASP. In *Proc. of the 15th Int. Conf. on Computer Aided Verification (CAV'03)*, volume 2725 of *LNCS*, pages 110–113. Springer-Verlag, 2003.
3. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.
4. E. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. In *Proc. of the 6th Int. Conf. on Computer Aided Verification (CAV'94)*, volume 818 of *LNCS*, pages 415–427. Springer-Verlag, 1994.
5. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
6. R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.
7. P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *Proc. of the 16th Int. Conf. on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004.

8. W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *Proc. of the 9th Int. SPIN Workshop (SPIN'02)*, volume 2318 of *LNCS*, pages 95–111. Springer-Verlag, 2002.
9. W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.
10. X. Huang, C. Luo, and R. van der Meyden. Improved bounded model checking for a fair branching-time temporal epistemic logic. In *Proc. of 6th Int. Workshop on Model Checking and Artificial Intelligence 2010*, LNAI. Springer, 2011.
11. A. V. Jones and A. Lomuscio. Distributed bdd-based bmc for the verification of multi-agent systems. In *AAMAS*, pages 675–682, 2010.
12. M. Kacprzak, A. Lomuscio, A. Niewiadomski, W. Penczek, F. Raimondi, and M. Szreter. Comparing BDD and SAT based techniques for model checking Chaum's dining cryptographers protocol. *Fundam. Inform.*, 72(1-2):215–234, 2006.
13. M. Kacprzak, A. Lomuscio, and W. Penczek. From bounded to unbounded model checking for temporal epistemic logic. *Fundam. Inform.*, 63(2-3):221–240, 2004.
14. Alessio Lomuscio, Wojciech Penczek, and Hongyang Qu. Partial order reduction for model checking interleaved multi-agent systems. In *AAMAS, IFAAMAS Press.*, pages 659–666, 2010.
15. R. van der Mayden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *Proc. of the 17th IEEE Computer Security Foundations Workshop (CSFW-17)*, pages 280–291. IEEE Computer Society, June 2004.
16. A. Męski, W. Penczek, and M. Szreter. Bounded model checking linear time and knowledge using decision diagrams. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'11)*, pages 363–375, 2011.
17. A. Męski, W. Penczek, M. Szreter, B. Woźna-Szcześniak, and A. Zbrzezny. Bounded model checking for knowledge and linear time. In *Proceedings of the 11th AAMAS*. IFAAMAS Press, 2012. To appear.
18. R. van der Meyden and N. V. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proc. of the 19th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, volume 1738 of *LNCS*, pages 432–445. Springer-Verlag, 1999.
19. D. Peled. All from one, one for all: On model checking using representatives. In *Proc. of the 5th Int. Conf. on Computer Aided Verification (CAV'93)*, volume 697 of *LNCS*, pages 409–423. Springer-Verlag, 1993.
20. W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundam. Inform.*, 55(2):167–185, 2003.
21. W. Penczek, B. Woźna-Szcześniak, and A. Zbrzezny. Towards SAT-based BMC for LTLK over interleaved interpreted systems. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'11)*, pages 565–576, 2011.
22. F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic*, 5(2):235–251, 2007.
23. K. Su, Abdul Sattar, and Xiangyu Luo. Model checking temporal logics of knowledge via OBDDs. *The Computer Journal*, 50(4):403–420, 2007.

Preface of the International Workshop on Petri Net-based Security (WooPS)

Rafael Accorsi[1], Tadao Murata[2], and Silvio Ranise[3]

[1] University of Freiburg, Germany
accorsi@iig.uni-freiburg.de
[2] University of Illinois at Chicago, USA
murata@uic.edu
[3] Fundazione Bruno Kessler, Italy
ranise@fbk.eu

Petri nets provide an expressive and well-studied formalism to specify and reason about security and reliability properties of abstractions/views of modern applications, such as smart grids and distributed enterprise systems. It is however unclear how the cornucopia of available techniques (based on Petri nets and related concurrency models) can provide results of the analysis with sufficient precision to increase the confidence of designers in the overall security of the new applications. The assessment of these techniques with respect to correctness, computational complexity, and real-world case-studies is indeed mandatory to significantly advance the state-of-the-art of the emerging research area.

The WooPS session was centered around the study of new methodologies and analysis techniques at the crossroads of the Petri net and security communities.

Concerning methodologies, in the paper "Analysing SONAR Model Transformations," Köhler-Bussmeier studies the dynamics of organization models, described as Petri nets, under model transformations, specified as transitions of a Petri net. With regard to analysis techniques, in the paper "Inference of Local Properties in Petri Nets Composed through an Interface," Ferigato and Mangioni study a notion of visibility of the local states for composition of elementary Petri nets representing a service provider (defender), a client of the service (attacker), and the protocol of interaction (interface). A notion of visibility formalizes the idea that an attacker tries to infer the validity of a local state of the defender even though only the interface is observable.

We would like to thank the authors of the papers for their efforts in producing stimulating contributions to the WooPS session. We also thank Prof. Karsten Wolf for his instigating keynote address during the WooPS session. Finally, thanks to the members of the organizing committees of the 33rd International Conference on Application and Theory of Petri Nets and Concurrency and the 12th International Conference on Application of Concurrency to System Design to which this event is affiliated.

Rafael Accorsi, Tadao Murata, and Silvio Ranise

Developing and Integrating Petri net tools - an Experience Report (Invited Talk)

Karsten Wolf

Universität Rostock
Institut für Informatik

Academic tools are increasingly important outcomes of academic research. They can be used for proving applicability of theoretical results, for linking theory and practice, and they can serve as well assessable deliverables of research projects. The Petri net group in Rostock has a long tradition in developing Petri net related academic tools. Today, about 20 tools are contributed, developed by teams working in different cities and countries. Our tools have been integrated into several general verification frameworks.

In the talk, we would like to give an overview on tool development in Rostock and discuss our general tool philosophy, including interoperability and integration. We will also share our experience concerning general issues in academic tool development as opposed to industrial software engineering. Our examples will include recent tools such as the Anica tool that checks an information flow security property on Petri nets.

Analysing SONAR Model Transformations

Michael Köhler-Bußmeier

University of Hamburg, Department for Informatics
Vogt-Kölln-Str. 30, D-22527 Hamburg
koehler@informatik.uni-hamburg.de

Abstract. In this paper we study the space of organisation models that are reachable via model transformation in our SONAR-framework.
The space of organisation models is defined as a Petri net, where each reachable marking represents one SONAR-organisation model and each transitions represent a model transformation.
Keywords: multi-agent systems, organisation centred design, model transformation, Petri net, SONAR

1 Introduction

In virtual enterprises different partners, called agents, cooperate within an organisational setting. A major research focus for multi-agent systems (MAS) is the coordination of self-interested agents. Recent work puts emphasis on the organisation, that enables the teamwork of agents (cf. [1, 2] for an overview). Teamwork includes aspects like team formation, team planing (distributed problem solving), and coordinated plan execution.

The interdisciplinary field *socionics* [3], situated between sociology and computer science, emphasises the *interplay* of the macro level (i.e. the organisation) and the micro level (i.e. the agent), i.e. an organisation is not only a passive structure that provides a guiding frame for the teamwork – with the same right one could define it the other way around: The organisation is the dynamic entity that evolves in the context of the interaction of agents. In analogy to the notion of *teamwork* we like to coin this dynamics as *orgwork*.

Our in-depth discussion of the interplay of agents and organisations shows that both are active entities (cf. [4]), which influence each other at the same time. The conceptual background is a little bit different from the mainstream in organisation-centred MAS: While both agree on the fact that organisations are entities that are not static, but evolve, organisation-centred MAS motivate this aspect from the desire to allow some kind of adaption at the organisation level, while in socionics agents and organisations are instantiations of the *same* concept, which naturally implies that organisation plan, learn, adapt, etc. like agents do. Since agents and organisations are essentially the same, we are free to describe a system either from the agent-perspective, from the organisation-perspective, or from their interplay-perspective. We will concentrate on the interplay-perspective in the following, i.e. the orgwork.

In this presentation we show how the orgwork is modelled in our SONAR-framework (short for: Self-Organising Net ARchitecture). The teamwork aspects of SONAR have been studied in [5]. The orgwork aspects of SONAR are devoted to *distributed* model transformations. Note, that in SONAR teamwork and orgwork are entangled, i.e. model transformations are not independent from agent interactions – they are the other side of the coin. Each teamwork is an orgwork, as it generates a transformation. From the organisation perspective one could say that the organisation *learns* during the model transformation.

We already have a rich theoretical basis for the teamwork, which is based on Petri nets [6]: Teams are unfoldings of delegation nets, plans are unfoldings of multi-party workflow nets, etc. In this paper, we demonstrate that model transformations could also be handled within the Petri net theory. For this purpose, we introduce so called *meta-organisation nets*. The marking of a meta-organisation net describes an organisation model, the firing of a meta-transition describes a model transformation. This might seem a little bit unusual, since most approaches specify model transformations within a graph rewriting context [7]. Here, we advocate for a Petri net based approach due to the following reasons: (i) We like to have only one type of formalism in SONAR to obtain an integrated, lean formal setting and (ii) we like team- and orgwork to be executed by the same engine (here: RENEW). It turns out, that many interesting properties of transformations could be formulated as natural net properties. Therefore, we could rely on well established analysis tools to investigate the space of all transformations.

The paper has the following structure: Section 2 introduces the SONAR-framework. Section 3 presents how the team-/org-work is generated from a SONAR-model. Analogously, Section 4 defines meta-organisation nets, which are used to specify the org-work. In Section 5 we define a simple logic to define organisation policies, i.e. properties, which have to be fulfilled by an organisation model and have to be preserved by the transformations. We show how meta organisation nets can be analysed to check properties of the space of organisation transformations.

2 The SONAR Framework

In this section we give a short introduction into our modelling formalism, called SONAR. A SONAR-model encompasses (i) a data ontology, (ii) a set of interaction models (called *distributed workflow nets*), (iii) a model, that describes the team-based delegation of tasks (called *role/delegation nets*), (iv) a network of organisational positions, and (v) transformation rules.

2.1 Distributed Workflows, Roles and Services

In the following we ignore the colour of of workflow tokens and restrict ourselves to black tokens and skip the discussion of the data ontology. In the following we fix a set of roles $\mathcal{R}$. A *distributed workflow net* (DWFN) $D = (P, T, F, r : T \to \mathcal{R})$ is a multi-party version of the well-known workflow nets [8] where the parties

are called *roles*. Each transition of a distributed workflow net is mapped by r to a role with the meaning that a transition t can executed only by an agent that implements the role $r(t)$.

Let $R(D)$ be the set of roles used by D, i.e. $R(D) := r(T_D)$. For each set of roles $R \subseteq R(D)$ we can construct the subnet $D[R] = (P_R, T_R, F_R)$ of $D = (P_D, T_D, F_D)$ (called the role-component generated by R) by setting $T_R := r^{-1}(R)$, $P_R := ({}^\bullet T_R \cup T_R{}^\bullet)$, and $F_R := F_D \cap (P_R \cup T_R)^2$. All message places become places at the border of $D[R]$. Each partition $R_1, \ldots, R_k$ on the set of roles in D also decomposes D into its role-components: $D = D[R_1] \| \ldots \| D[R_k]$. The role-component $D[R]$ defines the *service* provided by D w.r.t. the roles R. For singletons $D[\{r\}]$ we write $D[r]$.

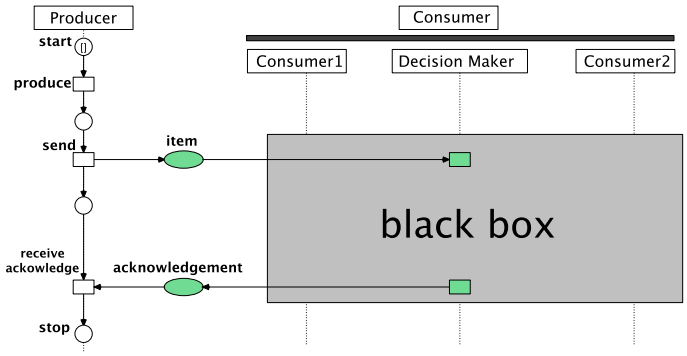


Fig. 1. Refined Distributed Workflow

$D[R] \simeq D'[R']$ denotes the fact that a component $D[R]$ cannot distinguished from another component $D'[R']$ with the same interface. This is formalised as a bisimulation with respect to the input/output behaviour at the message interface. The DWFN PC_2 in Figure 1 shows such a refinement, where the role *Consumer* of another DFWN PC (not shown here) has been refined by the interaction of the three roles $Consumer_1$, *Decision Maker*, and $Consumer_2$. The fact that the consumer part $PC[Consumer]$ is i/o-bisimlar to the part $PC_2[Consumer_1, Decision\ Maker, Consumer_2]$ is denoted:

$$PC[Consumer] \simeq PC_2[Consumer_1, Decision\ Maker, Consumer_2]$$

Let $\mathcal{D}$ be a set of DWF nets. Then, $(\mathcal{R}, \mathcal{D}, \simeq)$ is called a DWFN *repository*.

2.2 The Formal Organisation

Assume that $\mathcal{A}$ is the set of agents. On the conceptual level we define the tasks the organisation is responsible for and how they are handled. In SONAR, organisation is a net, where each place is of the form $p = \mathsf{task}^a_{D[R]}$, which describes a

task for the agent a to establish the service $D[R]$. Each transition t is either a delegation, a split, a refinement, or an execution operation (cf. Fig. 2):[1]

1. Delegate: The task to implement DWFN $D[r]$ is delegated from agent a to b. Only the delegation operation delegates the ownership of a task.
2. Split: The DWFN $D[r_1, \ldots, r_n]$ is split into the component $D[r_1], \ldots, D[r_n]$. Note, that this operation does not alter the interaction behaviour since $D[r_1, \ldots, r_n] \simeq (D[r_1] \| \cdots \| D[r_n])$.
3. Refinement: The DWFN $D[r]$ is replaced by $D'[r_1, \ldots, r_n]$, which has to be a refinement, i.e. $D[r] \simeq D'[r_1, \ldots, r_n]$ must hold.
4. Execution: The DWFN $D[r]$ is executed by the agent that is responsible for the task.

The set of all tasks and operations is defined as follows:

$$\mathcal{P} := \{\mathsf{task}^a_{D[R]} \mid D \in \mathcal{D} \wedge R \subseteq R(D), a \in \mathcal{A}\}$$

$$\mathcal{T}_{deleg} := \left\{\mathsf{d}(\{\mathsf{task}^a_{D[r]}\}, \{\mathsf{task}^b_{D[r]}\}) \;\middle|\; D \in \mathcal{D} \wedge r \in R(D) \wedge a, b \in \mathcal{A} \wedge a \neq b\right\}$$

$$\mathcal{T}_{split} := \left\{\mathsf{s}(\{\mathsf{task}^a_{D[r_1,\ldots,r_n]}\}, \{\mathsf{task}^a_{D[r_1]}, \ldots, \mathsf{task}^a_{D[r_n]}\}) \;\middle|\; D \in \mathcal{D} \wedge \{r_1, \ldots, r_n\} \subseteq R(D) \wedge n > 1 \wedge a \in \mathcal{A}\right\}$$

$$\mathcal{T}_{refine} := \left\{\mathsf{r}(\{\mathsf{task}^a_{D[r]}\}, \{\mathsf{task}^a_{D'[R]}\}) \mid D, D' \in \mathcal{D} \wedge D \neq D' \wedge D[r] \simeq D'[R] \wedge a \in \mathcal{A}\right\}$$

$$\mathcal{T}_{exec} := \left\{\mathsf{e}(\{\mathsf{task}^a_{D[r]}\}, \emptyset) \mid D, D' \in \mathcal{D} \wedge a \in \mathcal{A}\right\}$$

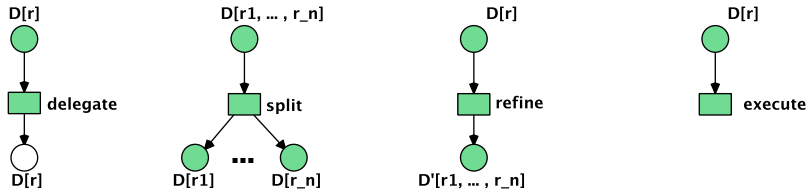


Fig. 2. Delegation, Split, Refinement, and Excution

We define $\mathcal{T} := \mathcal{T}_{deleg} \cup \mathcal{T}_{split} \cup \mathcal{T}_{refine} \cup \mathcal{T}_{exec}$. Note, that the sets $\mathcal{T}_{deleg}$, $\mathcal{T}_{split}$, $\mathcal{T}_{refine}$, and $\mathcal{T}_{exec}$ are pairwise disjoint.

Let $t = \mathsf{op}(X, Y) \in \mathcal{T}, \mathsf{op} \in \{d, s, r, e\}$, then we define the flow relation $\mathcal{F}$ by ${}^\bullet t = X$ and $t^\bullet = Y$,

The mapping $\alpha : \mathcal{P} \to \mathcal{A}$ returns the owner of a task: $\alpha(\mathsf{task}^a_{D[R]}) := a$. For each t we define its ownership equal to the owner of the place in the preset. Then all conflicts are agent-internal: $\forall p \in P : \forall t \in p^\bullet : \alpha(t) = \alpha(p)$.

[1] This is a slight modification of the definition in [6, 5], which allows that a transition t is a delegation, split, and refinement at the same time. We have chosen this simplified version for presentational purposes.

The places $\mathcal{P}$ and transitions $\mathcal{T}$ encode implicitly several aspects: the operation type, the ownership of tasks, and the agency of operations.

A SONAR-model is stratified in the sense that each node of organisation net belongs to a specific level n. Assume we have $(\mathcal{P}, \mathcal{T}, \mathcal{F})$ as given above. We assume that each DWFN $D \in \mathcal{D}$ belongs to exactly one level $n = n(D)$ and for each $p = \mathsf{task}^a_{D[R]} \in \mathcal{P}$ we set its level to $n(p) = n(D)$. Let $\mathcal{P}_n$ be the set of all place with level n. Furthermore, we assume that for each $t \in \mathcal{T}$ the level does not change, i.e. all the places p in ${}^\bullet t \cup t^\bullet$ belong to the same level. We define the level of $t \in \mathcal{T}$ as the level of the surrounding places. Let $\mathcal{T}_n$ be the set of all place with level n.

An *organisation* is a Petri net that contains some transitions of $\mathcal{T}$ and for each transition the complete pre- and postset.

Definition 1. *A* organisation *is a Petri net $N = (P, T, F)$ with $T \subseteq \mathcal{T}$, $P = {}^\bullet T \cup T^\bullet$, and $F = \mathcal{F} \cap (P \cup T)^2$.*

The places in $P^0 := {}^\circ P := \{p \in P \mid {}^\bullet p = \emptyset\}$ are those tasks that the organisation is responsible for, i.e. tasks that are generated externally.

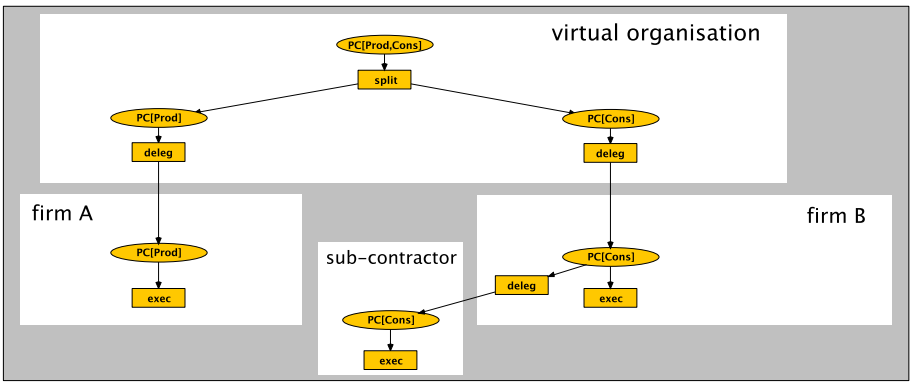


Fig. 3. An Example Organisation Net

Figure 3 shows an example organisation net, where the agents (*firm A, firm B* etc.) are indicated by the named boxes.

The stratification partitions the sets $\mathcal{P}$ and $\mathcal{T}$, i.e. $\mathcal{P} = \bigcup_{n\in\mathbb{N}} \mathcal{P}_n$ and $\mathcal{T} = \bigcup_{n\in\mathbb{N}} \mathcal{T}_n$. Similarily, each organisation $N = (P, T, F)$ is decomposed into $N := \bigcup_{n\in\mathbb{N}} N_n$, where each $N_n = (P \cap \mathcal{P}_n, T \cap \mathcal{T}_n, F \cap (\mathcal{P}_n, \cup\mathcal{T}_n)^2)$ is the organisation of level n.

The transitions in T are those operations that are explicitly allowed by the organisation N. This does not mean that the operations in $\mathcal{T} \setminus T$ are forbidden – some operations may become allowed whenever the organisation transforms. To specify which operations are permitted or forbidden we define organisation policies in Section 5.

2.3 Basic Transformations

We define two basic transformations: add_t and del_t where add_t adds the transition $t \in \mathcal{T}$ to the organisation $N = (P, T, F)$ provided that the preset of t is already part of N. Analogously, del_t removes t and its postset:

$$\mathsf{add}_t(P,T,F) := \begin{cases} (P \cup t^\bullet, T \cup \{t\}, F \cup {}^\bullet t \times \{t\} \cup \{t\} \times t^\bullet) & \text{if } t \notin T \wedge {}^\bullet t \subseteq P \\ \text{undef.} & \text{otherwise} \end{cases}$$

$$\mathsf{del}_t(P,T,F) := \begin{cases} (P \setminus t^\bullet, T \setminus \{t\}, F \setminus ({}^\bullet t \times \{t\} \cup \{t\} \times t^\bullet)) & \text{if } t \in T \\ \text{undef.} & \text{otherwise} \end{cases}$$

Then $ATF := \{\mathsf{add}_t, \mathsf{del}_t \mid t \in \mathcal{T}\}$ is the set of all atomic transformations. A transformation is the composition $\tau = \tau_1; \cdots ; \tau_n$ of atomic transformations.

The *level* of a basic transformation add_t or del_t is defined as the level of t, i.e. $n(\mathsf{add}_t) = n(t)$ and $n(\mathsf{del}_t) = n(t)$.

We now come to the org-work part: Each transition t_D of a DWFN D is labelled with a basic transformation: $\lambda(t_D) = \mathsf{add}_t$ or $\lambda(t_D) = \mathsf{del}_t$ – or with $\lambda(t_D) = \bot$ to indicate the absence of a transformation. The intended meaning is that the execution of the DWFN transition t_D also executes the basic transformation $\lambda(t_D)$. We will come back to this point in Section 3.

We require that the transformation inscription of each transition in D has a level less than the level $n(D)$ of the DWFN D itself. Therefore, each firing sequence $w = t_1 \cdots t_n$ of the DWFN generates a sequence of transformations $\lambda(w) = \lambda(t_1) \cdots \lambda(t_n)$ and due to the level restriction on the $\lambda(t_i), i = 1..n$ we obtain the property that the DWFN transforms only lower organisation levels.

3 Team-Work and Org-Work

In the following we give a short explanation of the teamwork derived from a SONAR-model: team formation, the team-DWFN, team planning via negotiation, and organisational transformations as show in Fig. 4. This is just a short summary - cf. [5] for details.

The processes described below are implemented by a specific middleware, called MULAN4SONAR, which is parametrised by a concrete SONAR-model. The generic part of the MULAN4SONAR-middleware is specified by a high-level Petri net, namely a reference net. This is beneficial for two reasons: (1) the prototype directly incorporates the main Petri net structure of the SONAR-model; (2) the prototype is immediately functional as reference nets are directly executable using the open-source Petri net simulator RENEW [9] and we can easily integrate the prototype into MULAN [10, 11], our development and simulation system for MAS based on Java and reference nets.

Team Formation: Processes of the Organisation Net Team formation for a given task $\mathsf{task}^a_{D[R]}$, i.e. the assignment to a to implement $D[R]$, is then expressed as an execution sequence w from the initial marking $m_0 = \mathsf{task}^a_{D[R]}$

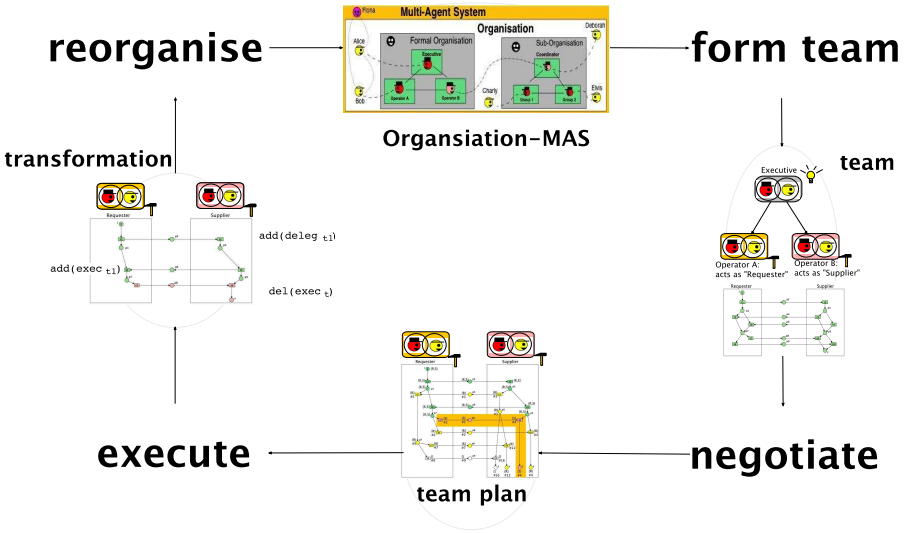


Fig. 4. Interplay of team- and org-work in SONAR

to the empty marking $m = \mathbf{0}$, i.e. $m_0 \xrightarrow{w} \mathbf{0}$ assigns an executing agents to each (sub-)tasks p.

Note, that the following refinement property holds for all task assignments: Whenever we put a token on the place $\mathsf{task}^a_{D[R]}$ then each reachable marking m describes a refinement of $D[R]$.

If Petri net processes (i.e. partially ordered runs) are used instead of sequences, then the net structure of the process can be used as the team's interaction structure (for the formal definitions of "teams-as-processes" cf. [6]). Here, only maximal processes are considered (i.e. we assume the progress property).

Assume that we have an organisation team OT, i.e. a process of the organisation net for the initial marking $m_0 = \mathsf{task}^a_{D[R]}$ together with the process morphism ϕ. For each reachable place-cut C of OT define the DWFN $D(C)$ as the composition of the role-components of the final transitions:

$$D(C) := \|_{b \in C} D(\phi(b))[R(\phi(b))]$$

Since each transition in the organisation N actually refines $D[R]$, we obtain that for each reachable place-cut C in a team OT the distributed workflow net $D(C)$ is a refinement of $D[R]$: $D[R] \simeq D(C)$.

The Team-DWFN Each team OT generates the team-DWFN $D(OT)$ that is derived from the executing transitions, which are the maximal transitions in the process net.

Each group member starts with its individual partial plan, which is an unfolding of the team-DWFN $D(OT)$ with the property that all different branches

of the unfolding lead to the final state. The set of all partial plans of a workflow N is denoted $\mathbb{PP}(N)$.

Assume that we have the team OT and its team-DWFN $D(OT)$. From OT we can deduce which agents implements which role of the team-DWFN: $\alpha_{OT} : R(D(OT)) \to \mathcal{A}$ maps each role r in the team-DWFN to the agent of the team that is assigned to r.

Team Planning: Group Plan Negotiation Each team generates a *team-plan* via negotiation. In our formal setting, Petri net unfoldings are used since a compromise can be easily characterised in terms of intersection of Petri nets – which is not that easy for a *sequential* formalisation of partial plans.

A team OT defines a tree-like structure, i.e. a team consists of sub-team, etc. Note, that the same agent can occur several times at different positions within a team. We denote this tree as nested sets, which we denote by G_{OT}.

Let G be some sub-group, i.e. a subset of G_{OT}. During the negotiation each group member $g \in G$ (which is a group again) recursively calculates its local partial plan π_g. The intersection $\bigcap_{g \in G} \pi_g$ of all these partial plans is not a partial plan in general: If all group members reach the final state of the workflow via different processes the intersection does not contain the final state at all.

For each G we define a *group-plan* π_G as a partial-plan with minimal distance to the intersection $\bigcap_{g \in G} \pi_g$ of all the local plans: Let $\mathbb{PP}(N, (\pi_g)_{g \in G}, d)$ denote the set of all partial plans such that we have to expand the intersection $\bigcap_{g \in G} \pi_g$ by at most d nodes.

The negotiation protocol is roughly the following: The hierarchical structure of the team G induces a the set of sub-groups. The negotiation protocol selects a sub-team G' and an initial distance d. Then the agents within G' compute a non-empty approximating subset of $\mathbb{PP}(N, (\pi_g)_{g \in G'}, d)$. Iteratively, these sets of group-plans are combined to obtain an approximation for a "bigger" sub-group. It is allowed to extend the distance d whenever this seems appropriate.

Finally, the group $G = G_{OT}$ is considered and we obtain the team-plan $\pi_{OT} := \pi_{G_{OT}}$ for the whole team.

Group Plan Effect: Transformations Each group plan π_{OT} induces a transformation $\lambda(\pi_{OT})$ on the organisation. Therefore, we have a dynamics of the organisation, i.e. org-work.

Each sequence $w = t_1 \cdots t_n$ of transition in the team-DWFN $D(OT)$ generates the organisation transformation $\lambda(w) : ORG \to ORG$:

$$\lambda(w) := \lambda(t_1); \ldots; \lambda(t_n) := \lambda(t_n) \circ \cdots \circ \lambda(t_1)$$

We have already seen that due to the level restrictions on each $\lambda(t)$ the transformation $\lambda(w)$ transforms only *lower* levels of the organisation.

There is another constraint for transformations that arises from the ownership within the team OT: From a given team OT we know the agent $a = \alpha_{OT}(r(t_D))$ that executes the team-DWFN transition t_D in the team plan. In

general, the executor can be different from the owner of a transformation, where the owner is the agent that owns the manipulated t: $\alpha(\mathsf{add}_t) := \alpha(t)$ and $\alpha(\mathsf{del}_t) := \alpha(t)$. But since agents are autonomous, agents cannot "manipulate" each other. If one agents likes to transform t (i.e. add or delete it), but does not own it, it has to negotiate with the owner about it. Therefore, we require that the executor $\alpha_{OT}(r(t_D))$ of a transformation $\lambda(t_D)$ is also its owner $\alpha(\lambda(t_D))$:

$$\forall t_D \in T_D : \lambda(t_D) \neq \bot \Longrightarrow \alpha_{OT}(r(t_D)) = \alpha(\lambda(t_D))$$

Definition 2. *Assume that N is an organisation and OT is a team.*

*A firing sequence $w = t_1 \cdots t_n$ of the team-*Dwfn $D(OT)$ *is a* team-transformation *if each transformation is executed by the agent that owns it:* $\forall 1 \leq i \leq n : \lambda(t_D) \neq \bot \Longrightarrow \alpha_{OT}(r(t_i)) = \alpha(\lambda(t_i))$.

A firing sequence is called applicable to the organisation N *if the transformation $\lambda(w)$ is defined for N.*

We can extend these notions to partially ordered runs of a Dwfn. Whenever we have two concurrent events e_1 and e_2 in the run, then we require that the transformations $\lambda(\phi(e_1))$ and $\lambda(\phi(e_2))$ are independent and owned by the right agent.

4 Formalisation of Org-Work by Meta-Organisations

The main problem of the negotiation process is to ensure that the transformation generated by a team plan is applicable to the current organisation. The approach taken here is to define *meta-orgaisations*. A meta-organisation net encodes in its marking the current organisation and can "decide" which transformations (i.e. add_t and del_t) are currently applicable. Therefore, a firing sequence of the team-Dwfn is applicable iff it is enabled in the meta-organisation.

We can guarantee that the transformation generated by the team plan is applicable to the organisation if we *synchronise* the team-Dwfn $D(OT)$ with the meta-organisation $\hat{N}_{P^0}$. And during the negotiation, we do not construct a partial plan for the team $D(OT)$, but for the synchronous product of $D(OT)$ and the meta-organisation $\hat{N}_{P^0}$. Then each team plan (more precisely: the subnet that is obtained by restricting the process to nodes belonging to the team-Dwfn $D(OT)$) fulfils the transformation constraints of Def. 2 by construction.

Here, we see the benefit to have an integrated model for team- and org-work: Both are Petri nets, which allows a very simple definition of the synchronisation as a net product.

In the following we define transformations as processes of another Petri net $\hat{N}_{P^0}$, called the *meta-organisation*. Assume a fixed universe $(\mathcal{P}, \mathcal{T}, \mathcal{F})$. We also assume a set of places $P^0 \subseteq \mathcal{P}$ that contains all tasks that the organisation is responsible for.

For each $n \in \mathbb{N}$ we define the *meta-organisation of level n* as $\hat{N}_n = (\hat{P}_n, \hat{T}_n, \hat{F}_n)$ to describe the possible transformation processes.

– We define the set of meta-places and the set of meta-transitions:

$$\hat{P}_n := \{\hat{p} \mid p \in \mathcal{P}_n\} \cup \{\mathsf{on}_t, \mathsf{off}_t \mid t \in \mathcal{T}_n\}$$
$$\hat{T}_n := \{\mathsf{activate}_t, \mathsf{deactivate}_t \mid t \in \mathcal{T}_n\}$$

– The meta-arcs for $\mathsf{activate}_t$ are defined by:

$${}^\bullet\mathsf{activate}_t := \{\hat{p} \mid p \in {}^\bullet t\} \cup \{\mathsf{off}_t\} \quad \text{and} \quad \mathsf{activate}_t{}^\bullet := \{\hat{p} \mid p \in {}^\bullet t \cup t^\bullet\} \cup \{\mathsf{on}_t\}$$

A token on $\hat{p}$ is used to "activate" each transformation on $t \in p^\bullet$, i.e. each transition $\mathsf{activate}_t$.
Each transition $\mathsf{deactivate}_t$ reverts the activation transition:

$${}^\bullet\mathsf{deactivate}_t := \mathsf{activate}_t{}^\bullet \quad \text{and} \quad \mathsf{deactivate}_t{}^\bullet := {}^\bullet\mathsf{activate}_t$$

– Since all the $\mathcal{P}_n$ and $\mathcal{T}_n$ are disjoint, so are the meta-organisations $\hat{N}_n$ and we can define their union, too: $\hat{N}_{P^0} := \bigcup_{n\in\mathbb{N}} \hat{N}_n := \left(\bigcup_{n\in\mathbb{N}} \hat{P}_n, \bigcup_{n\in\mathbb{N}} \hat{T}_n, \bigcup_{n\in\mathbb{N}} \hat{F}_n\right)$.
– The initial meta-marking $\hat{m}_0$ marks all meta-places off_t and the task places $\hat{p}$ that the organisation is responsible for:

$$\hat{m}_0 := \{\hat{p} \mid p \in P^0\} \cup \{\mathsf{off}_t \mid t \in \mathcal{T}\}$$

Figure 5. shows the initial fragment of the meta-organisation $\hat{N}_{P^0}$ for the set $P^0 = \{\mathsf{task}^{O_1}_{PC[Prod,Cons]}\}$.

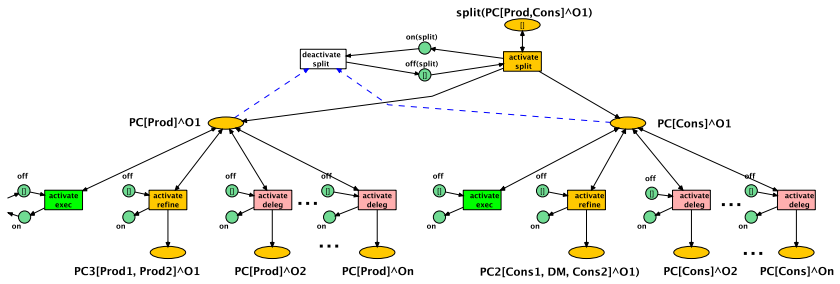


Fig. 5. Fragment of the Meta-Organisation $\hat{N}_{P^0}$ with $P^0 = \{\mathsf{task}^{O_1}_{PC[Prod,Cons]}\}$

The following proposition gives a characterisation of the reachable markings.

Proposition 1. *Assume* $\hat{m}_0 \xrightarrow{\hat{w}} \hat{m}$ *for some* $\hat{w} = \hat{t}_1 \cdots \hat{t}_n$.

1. *For all* $t \in \mathcal{T}$ *the places* on_t *and* off_t *are 1-safe, since* $\hat{m}(\mathsf{on}_t) + \hat{m}(\mathsf{off}_t) = 1$.
2. *For each* $\hat{p} \in \hat{P}$ *we have:* $\hat{m}(\hat{p}) = 1_{P^0}(p) + |I^+(\hat{w}, \hat{p})| - |I^-(\hat{w}, \hat{p})|$

where $I^+(\hat{w}, \hat{p}) := \{i \in \{1..n\} \mid \hat{t}_i = \mathsf{activate}_t \wedge t \in {}^\bullet p\}$
$I^-(\hat{w}, \hat{p}) := \{i \in \{1..n\} \mid \hat{t}_i = \mathsf{deactivate}_t \wedge t \in {}^\bullet p\}$.

Induced Organisation The meta-places on_t are used to "link" an organisation and its meta-organisation: We obtain an organisation by selecting those transitions t that are activated by a token on on_t.

Definition 3. *Let $\hat{N}_{P^0}$ be a meta-organisation and $\hat{m} \in RS(\hat{N}_{P^0}, \hat{m}_0)$ a reachable marking. The* organisation induced by $\hat{m}$ *is $N(\hat{m}) := (P_{\hat{m}}, T_{\hat{m}}, F_{\hat{m}})$, where $T_{\hat{m}} := \{t \in \mathcal{T} \mid \hat{m}(\mathsf{on}_t) = 1\}$, $P_{\hat{m}} = P^0 \cup {}^\bullet T_{\hat{m}} \cup T_{\hat{m}}{}^\bullet$, and $F_{\hat{m}} = \mathcal{F} \cap (P_{\hat{m}} \cup T_{\hat{m}})^2$.*

In SONAR, we use meta sequences $\hat{w}$ to encode organisations, since each firing sequence $\hat{w} \in \hat{T}_n^*$ starting in $\hat{m}_0$, i.e. $\hat{m}_0 \xrightarrow{\hat{w}} \hat{m}$, generates an organisation net in a natural way: $N(\hat{w}) := N(\hat{m})$.

The following shows that construction $N(\cdot)$ is injective:

Lemma 1. *Let N be an organisation net and $\hat{N}_{P^0}$ a meta-organisation. Whenever there is a meta-marking $\hat{m}$ such that $N = N(\hat{m})$ holds, then $\hat{m}$ is uniquely defined.*

Proof. Assume that there are two reachable meta-markings, say $\hat{m}_1$ and $\hat{m}_2$ with $\hat{m}_0 \xrightarrow{\hat{w}} \hat{m}_1$ and $\hat{m}_0 \xrightarrow{\hat{w}} \hat{m}_2$, that both generate N, i.e. we have: $N = N(\hat{m}_1) = N(\hat{m}_1)$.

We know the invariance $\hat{m}(\mathsf{on}_t) + \hat{m}(\mathsf{off}_t) = 1$ for all reachable markings $\hat{m}$. Assume that $\hat{m}_1$ and $\hat{m}_2$ differ on some $\mathsf{on}_t/\mathsf{off}_t$ pair. Then we have $\hat{m}_1(\mathsf{on}_t) = 1$ and $\hat{m}_2(\mathsf{on}_t) = 0$ (or vice versa) and $t \in T_{N(\hat{m}_1)}$, but $t \notin T_{N(\hat{m}_2)}$ and therefore $N(\hat{m}_1) \neq N(\hat{m}_2)$.

Since $\hat{m}_1$ and $\hat{m}_2$ agree on each $\mathsf{on}_t/\mathsf{off}_t$ pair, we know that for each t we have the same balance in $\hat{w}_1$ and $\hat{w}_2$:

$$|\hat{w}_1|_{\mathsf{activate}_t} - |\hat{w}_1|_{\mathsf{deactivate}_t} = |\hat{w}_2|_{\mathsf{activate}_t} - |\hat{w}_2|_{\mathsf{deactivate}_t}$$

Since this implies $|I^+(\hat{w}_1)| - |I^-(\hat{w}_1)| = |I^+(\hat{w}_2)| - |I^-(\hat{w}_2)|$, we know that $1_{P^0}(p) + |I^+(\hat{w}_1, \hat{p})| - |I^-(\hat{w}_1, \hat{p})| = 1_{P^0}(p) + |I^+(\hat{w}_2, \hat{p})| - |I^-(\hat{w}_2, \hat{p})|$ and by Prop. 1 we have $\hat{m}_1(\hat{p}) = \hat{m}_2(\hat{p})$ for each $\hat{p}$. □

The Meta-Organisation as a High-Level Net In MULAN4SONAR we model the meta organisation as a a high-level Petri net. Figure 6 shows the RENEW-based model for the meta organisation net. The place meta places contains all tokens of the form $\hat{p}$. The place actions contains all tokens of the form on_t. Due to the invariant $\hat{m}(\mathsf{on}_t) + \hat{m}(\mathsf{off}_t) = 1$ the marking of off_t can represented implicitly by looking for absent tokens of the form on_t. The transition add action is enabled whenever there is one token $\hat{p}$ on meta places such that ${}^\bullet t = \{p\}$ (which is specified by the inscription act = [type, task, tasks]) and *no* token of the form on_t on actions. Note, that the arc from actions to add action is an inhibitor arc. Whenever add action fires, it generates the action, i.e. puts the token on_t on the place actions and foreach $p \in t^\bullet$ one token $\hat{p}$ on meta places. Note, that the arc from add action to meta places is a so called flexible arc, which generates a multiset of flexible cardinality.

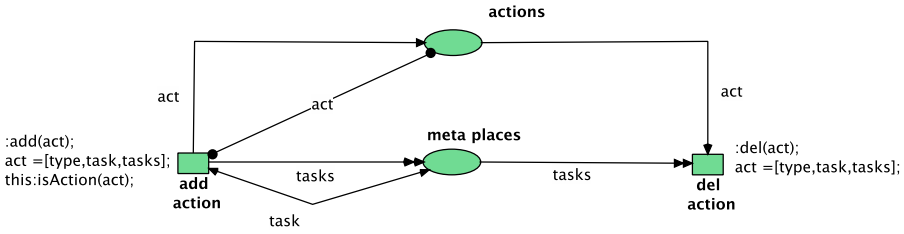


Fig. 6. The High-Level Net Variant for the Meta Organisation Net (Fragment)

Induced Transformations Each meta-transition induces a transformation. In the following we show that for $\hat{m}_1 \xrightarrow{\hat{w}} \hat{m}_2$ the organisation generated from a meta-marking, i.e. $N(\hat{m}_2)$, coincides with the organisation obtained from the induced transformation, i.e. $\tau(w)(N(\hat{m}_1))$.

The *transformation induced by the meta-transition* $\hat{t}$ is $\tau_{\hat{t}}$, which defined as:

$$\tau_{\hat{t}}(N) = \begin{cases} N(\hat{m}'), & \text{if } \exists \hat{m}, \hat{m}' : \hat{m} \in RS(\hat{N}_{P^0}, \hat{m}_0) \wedge N = N(\hat{m}) \wedge \hat{m} \xrightarrow{\hat{t}} \hat{m}' \\ \text{undef.}, & \text{otherwise} \end{cases}$$

Note, that Lemma 1 guarantees that $\tau_{\hat{t}}$ is well defined.

We extend the induced transformations to sequences of meta-transitions: Define $\tau_{(\hat{t}_1 \cdots \hat{t}_n)} := \tau_{\hat{t}_1}; \ldots; \tau_{\hat{t}_n}$ and $\tau_\epsilon = id$.

We can formalise the correspondence of the induced transformations and atomic transformations by defining the isomorphism $h : (\hat{T} \cup \{\bot\})^* \to ATF^*$ by

$$h(\mathsf{activate}_t) = \mathsf{add}_t, \quad h(\mathsf{deactivate}_t) = \mathsf{del}_t, \quad \text{and} \quad h(\bot) = id.$$

Each meta-sequence $\hat{w}$ induces a corresponding transformation:

Lemma 2. *Let $\hat{m}$ be a reachable marking. Each meta-sequence $\hat{w} = \hat{t}_1 \cdots \hat{t}_n$ induces a transformation $\tau_{\hat{w}}$ which coincides with $h(\hat{w})$ on $N(\hat{m})$:*

$$\hat{m} \xrightarrow{\hat{w}} \hat{m}' \quad \Longrightarrow \quad \tau_{\hat{w}}\ (N(\hat{m})) = h(\hat{w})\ (N(\hat{m}))$$

Proof. The general proposition follows by induction over the length over $\hat{w}$. The case $n = 0$ is clear, since $\tau_{\hat{\epsilon}} = id = h(\epsilon)$. The induction step follows form the definitions of the basic transformations add_t and and del_t: Whenever $\hat{t} = \mathsf{activate}_t$, then $\tau_{\hat{t}}$ is equivalent to add_t and whenever $\hat{t} = \mathsf{deactivate}_t$ then $\tau_{\hat{t}}$ is equivalent to del_t. □

Conversely, each transformation τ induces a corresponding meta-sequence:

Lemma 3. *Let the transformation $\tau = \tau_1; \ldots; \tau_n$ be defined for the organisation N and let $N = N(\hat{m})$ for some $\hat{m}$.*

Then τ induces the meta-sequence $h^{-1}(\tau)$ and $h^{-1}(\tau)$ is enabled in $\hat{m}$, i.e. $\hat{m} \xrightarrow{h^{-1}(\tau)} \hat{m}'$, and the generated organisations are equal: $\tau(N) = N(\hat{m}')$

Proof. Similar to the proof above. □

The relationship of transformations, organisations, and meta-organisation is illustrated by the following diagram:

$$\begin{array}{ccc} \hat{m} & \xrightarrow{\hat{w}} & \hat{m}' \\ \downarrow{\scriptstyle N(\cdot)} & & \downarrow{\scriptstyle N(\cdot)} \\ N = N(\hat{m}) & \xrightarrow{\tau_{\hat{w}}} & N' = N(\hat{m}') \end{array}$$

The product $D(OT) \otimes_{\alpha_{OT}} \hat{N}_{P^0}$ fuses each team-DWFN transition t_D with $\lambda(t_D) = \mathsf{add}_t$ with a copy of the meta-transition $\mathsf{activate}_t$ and each t_D with $\lambda(t_D) = \mathsf{del}_t$ with a copy of $\mathsf{deactivate}_t$.

5 Analysis of Organisation Transformations

In SONAR, policies are used to describe those properties that have to remain invariant during reorganisation processes of the model.

The set of atomic propositions is $AP = \{\mathbf{P}[t], \mathbf{O}[t], \mathbf{F}[t] \mid t \in \mathcal{T}\}$, where $\mathbf{P}[t]$ ($\mathbf{O}[t]$, $\mathbf{F}[t]$) means that it is *p*ermitted (*o*bligated, *f*orbidden) to perform the basic operation t.

A *policy* Φ is a propositional logic formula with AP as the set of atomic propositions. The set of all t occurring within the formula is denoted $\mathcal{T}_\Phi$.

Usually, it is not possible to permit and forbid a at the same time. Analogously, if there is an obligation to do t then t is usually permitted and not forbidden. Usually these constraints are encoded inside modal logic and the deontic qualifiers are modelled as modalities. For simplicity reasons, we use propositional logics and add a constraint for the truth assignment function instead.

Definition 4. *An* assignment *of a policy Φ is a mapping $v : AP \to \{0, 1\}$ with the property: $v(\boldsymbol{F}[t]) = 1 \implies v(\boldsymbol{P}[t]) = 0$ and $v(\boldsymbol{O}[t]) = 1 \implies v(\boldsymbol{P}[t]) = 1$ for all $t \in \mathcal{T}$. The set of all assignments is $ASSIGN$.*

We are interested in the fact, whether a organisation is a model for a policy.

Definition 5. *An organisation N is a* model *for a policy Φ (denoted $N \models \Phi$) whenever we have:*

$$\forall v \in ASSIGN : v(\Phi) = 1 \implies \big(\forall t \in \mathcal{T}_\Phi : \begin{array}{rcl} v(\boldsymbol{O}[t]) = 1 & \implies & t \in T \\ \wedge\, v(\boldsymbol{P}[t]) = 0 & \implies & t \notin T \\ \wedge\, v(\boldsymbol{F}[t]) = 1 & \implies & t \notin T \big) \end{array}$$

When we use a marked meta organisation $(\hat{N}_{P^0}, \hat{m})$ instead of N, then $t \in T$ in the definition above has to be replaced by $\hat{m}(\mathsf{on}_t) = 1$.

Analysis Each organisation transformation has to preserve the organisational policy, i.e. whenever $N \models \Phi$ holds and τ is a transformation, then $\tau(N) \models \Phi$ holds, too. Define the set of meta-markings that satisfy a policy Φ as:

$$SAT(\Phi) := \{\hat{m} \in RS(\hat{N}_{P^0}, \hat{m}_0) \mid N(\hat{m}) \models \Phi\}$$

Definition 6. *The meta-organisation* $\hat{N}$ enforces *the policy* Φ *if* $SAT(\Phi) = RS(\hat{N}_{P^0}, \hat{m}_0)$.

But in almost all cases this property does not hold, and this is not always problematic, since it is not necessary that each $N(\hat{m})$ satisfy Φ after each step of the team plan. It is only necessary that each $N(\hat{m})$ satisfy Φ at the *end* of the execution of the team plan.

We could formulate this property as follows: Assume that we have an initial model $N = N(\hat{m}_0)$ that satisfies the policy Φ and the meta-marking $\hat{m}$ is reachable, i.e. $N_1 = N(\hat{m})$ might be the result of a transformation. Whenever N_1 does not satisfy the policy, the question arises whether it is possible to reach a meta-marking $\hat{m}'$ such that $N_1 = N(\hat{m}')$ satisfies Φ:

$$\forall \hat{m} \in RS(\hat{N}_{P^0}, \hat{m}_0) : \exists \hat{m}' \in RS(\hat{N}_{P^0}, \hat{m}) : \hat{m}' \in SAT(\Phi)$$

Whenever this is not the case, then we known that there exist some transformation which should be suppressed in all team-plans, because we can never repair the situation. If the property holds, we know that each transformation can be extended to one that satisfies the policy again.

When understood as a question for the meta-organisation the answer is trivially "yes", since each transformation in $\hat{N}$ can be undone ad $\hat{N}$ is revertible. But of course it is undesired to reach a policy satisfying marking again, by undoing all transformations. Therefore we exclude some (or all) $\mathsf{deactivate}_t$ from the analysis: For $\hat{N} = (\hat{P}, \hat{T}, \hat{F})$ and $\hat{A} \subseteq \mathcal{P} \cup \mathcal{T}$ we define the subnet $(\hat{N} - \hat{A}) := (\hat{P} \cap \hat{B}, \hat{T} \cap \hat{B}, \hat{F} \cap \hat{B}^2)$, where $\hat{B} = (\mathcal{P} \cup \mathcal{T}) \setminus \hat{A}$.

Definition 7. *The meta-organisation* $\hat{N}$ *is* stable *w.r.t. the policy* Φ *if we have for* $\hat{A} = \{\mathsf{deactivate}_t \mid t \in \mathcal{T}\}$*:*

$$\forall \hat{m} \in RS(\hat{N} - \hat{A}, \hat{m}_0) : \exists \hat{m}' \in RS(\hat{N} - \hat{A}, \hat{m}) : \hat{m}' \in SAT(\Phi)$$

Proposition 2. *Assume, that* $\mathcal{A}$ *and* $\mathcal{D}$ *are finite sets.*

Given a meta-organisation $\hat{N}$*, it can be checked using standard model checking techniques, whether the policy is enforced and whether the policy is stable.*

Proof. Note, that whenever $\mathcal{A}$ and $\mathcal{D}$ are finite sets, then $\hat{N}$ is finite, too, and by Prop. 1 its state space is finite. Enforcement is a safety property and this can be checked doing an exhaustive state space exploration.

Stability is a kind of liveness property and can be checked by computing the strongly connected components (SCC) of the state space and checking whether each terminal SCC contains an $\hat{m}$ that satisfies Φ. □

6 Conclusion

In this paper we studied the organisation-oriented perspective of multi-agent systems and their transformations, which we named org-work as opposed to team-work.

We defined the dynamics of organisation models in the formalism of meta nets. This explicit representation of the history of the model transformation (understood by the meta-marking $\hat{m}$) allows us a deeper insight in the possible transformation that can arise as the byproduct of the teamwork.

Since we use the same formalism for team- and org-work, we can mix both models and obtain an integrated view on the system. The negotiation protocol directly benefits from this: During the negotiation, we do construct a partial plan not only for the team $D(OT)$, but for the synchronous product of $D(OT)$ and the meta-organisation $\hat{N}_{P^0}$. This guarantees that the transformation generated by a team plan π is *always* applicable to the actual organisation, which is a non-trivial property to ensure by negotiation.

References

1. Carley, K.M., Gasser, L.: Computational organisation theory. In Weiß, G., ed.: Multiagent Systems. MIT Press (1999) 229–330
2. Dignum, V., ed.: Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models. IGI Global (2009)
3. Malsch, T.: Naming the unnamable: Socionics or the sociological turn of/to distributed artificial intelligence. Autonomous agents and multi-agent systems **4** (2001) 155–186
4. Köhler, M., Moldt, D., Rölke, H., Valk, R.: Linking micro and macro description of scalable social systems using reference nets. In Fischer, K., Florian, M., Malsch, T., eds.: Socionics: Sociability of Complex Social Systems. Volume 3413 of LNAI, (2005) 51–67
5. Köhler-Bußmeier, M., Wester-Ebbinghaus, M., Moldt, D.: A formal model for organisational structures behind process-aware information systems. Transactions on Petri Nets and Other Models of Concurrency. **5460** (2009) 98–114
6. Köhler, M.: A formal model of multi-agent organisations. Fundamenta Informaticae **79** (2007) 415 – 430
7. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of algebraic graph transformation. Springer-Verlag (2006)
8. Aalst, W.v.d.: Verification of workflow nets. In Azeme, P., Balbo, G., eds.: Application and theory of Petri nets. Volume 1248 of LNCS (1997) 407–426
9. Kummer, O. et al.: An extensible editor and simulation engine for Petri nets: Renew. In Cortadella, J., Reisig, W., eds.: ATPN 2004. Volume 3099 of LNCS (2004) 484 – 493
10. Köhler, M., Moldt, D., Rölke, H.: Modeling the behaviour of Petri net agents. In Colom, J.M., Koutny, M., eds.: ATPN 2001. Volume 2075 of LNCS (2001) 224–241
11. Cabac, L., Dörges, T., Duvigneau, M., Moldt, D., Reese, C., Wester-Ebbinghaus, M.: Agent models for concurrent software systems. In Bergmann, R., Lindemann, G., eds.: MATES'08. Volume 5244 of LNAI (2008) 37–48

Inference of Local Properties in Petri Nets Composed through an Interface

Carlo Ferigato[1] and Elisabetta Mangioni[2]

[1] Joint Research Centre of the European Commission
via Enrico Fermi, 1, I-21027 Ispra, Italia
carlo.ferigato@jrc.ec.europa.eu
[2] DISCo - Università degli Studi di Milano-Bicocca
viale Sarca, 336, I-20126 Milano, Italia
mangioni@disco.unimib.it

Abstract. We study a notion of *visibility* of the local states of an Elementary Petri net obtained by composition through an interface. The components are three EN systems: the *defender*, providing a service to the environment, the *attacker*, a client of the service, and the *interface*, that models the protocol of interaction between the other two nets. Intuitively, the definition of *visibility* is meant to capture the idea that an *attacker* tries to infer the validity of a local state of the *defender* even if he can observe only the interface and itself. Our analysis is based on the notion of invariant properties and bisimilarity in Petri nets. We suggest also a measure of the degree of visibility of local states of the *defender* as seen by the *attacker*.

Keywords: Elementary Net System, composition, invariant

1 Introduction

The object of our study is open since the beginning of *Computer Science* [6]: we aim at a structural characterization of the hidden internal states of a system that become *visible* after its interaction with a defined subsystem. We assume to have a *high-level* system that wants to keep secret its internal local states from a *low-level* system interacting with the *high-level* component through an *interface*.

Basically, we explore the consequences of a proposal originally made in [3] for defining *non-interference* properties as *structural* properties by using the local validity of conditions as observable properties.

The general context of our study is known today in the literature as *non-interference*. The notions of *opacity* and *interference* between subsystems have been originally defined formally for *process algebras* [4]. In the context of Petri Nets, Busi and Gorrieri [3] applied these notions to Elementary Net Systems and Best, Darondeau and Gorrieri [2] extended recently the results to unbounded P/T Systems.

In these latter works, non-interference is basically defined as language equivalence. The equivalent languages are, respectively, the one generated by the restriction of the system to the *low-level* component alone, and the language generated by the composition of the *low-level* component with any *high-level* component.

The definition of non interference in terms of languages forces at considering *events* as basic observable entities, but this is partly in contradiction with the traditional view of events in nets as entities observable only indirectly, via the modifications of their pre- and post-conditions.

Since we consider as basic observables the local properties of systems represented by conditions, we call the property we describe *visibility*. In terms of *visibility*, two interacting systems can be seen as *defender* and *attacker*. The defender offers a service to the environment and wants to keep secret part of its local states. The attacker uses the service of the defender and tries to get information about its internal local states.

We will represent systems with Elementary Net (EN) systems, a basic model of Petri Nets. The service is modeled by a third EN system called *interface*. The interaction among these systems is given by the composition of the defender and the attacker through the interface. By using standard techniques related to S-invariants and bisimilarity in Petri Nets, we prove a theorem that allows us to recognize the places of the interface visible to, at least, one attacker. Moreover, we discuss the general cases of attackers bisimilar and non bisimilar to the interface. In the conclusions, we propose a measure of the *degree of visibility* of conditions as seen from the attacker.

2 Basic definitions

This section recalls basic definitions about net theory ([10]).

Definition 1. *An Elementary Net (EN) system is a quadruple $N = (B, E, F, m_0)$, where B and E are distinct finite sets of conditions and events, $F \subseteq (B \times E) \cup (E \times B)$ is the flow relation, $m_0 \subseteq B$ is the initial case and*

1. $dom(F) \cup ran(F) = B \cup E$.
2. $\forall e \in E, p, q \in B : (p, e), (e, q) \in F \Rightarrow p \neq q$

The preset of an element $x \in B \cup E$ is defined by ${}^\bullet x = \{y \in B \cup E | (y, x) \in F\}$; the postset of x is given by $x^\bullet = \{y \in B \cup E | (x, y) \in F\}$.

The structure of a net can be represented by a matrix M called the incidence matrix. In this matrix there is a row for each condition, a column for each event and the element (k, j) is set to 1 if there is an arc from the event e_j to the condition b_k, -1 if there is an arc from b_k to e_j, 0 otherwise.

The behaviour of EN systems is defined through the firing rule which specifies when an event can occur, and how event occurrences modify the holding of conditions. Let N be an EN system, $e \in E$ and $m \subseteq B$. The event e is *enabled* at m, denoted $m\,[e\rangle$, if ${}^\bullet e \subseteq m$ and $e^\bullet \cap m = \emptyset$; the occurrence of e at m leads

from m to m', denoted $m\,[e\rangle\,m'$, iff $m' = (m \setminus {}^\bullet e) \cup e^\bullet$. Let ϵ denote the empty word in E^*. It is possible to extend the firing rule to sequences of events in the following way:

$$m\,[\epsilon\rangle\,m$$

$$\forall e \in E, \forall w \in E^*, m\,[ew\rangle\,m' = m\,[e\rangle\,m'[w\rangle m''$$

and w is called *firing sequence*.

A subset $m \subseteq B$ is a *reachable marking* of N if there exists a $w \in E^*$ such that $m_0\,[w\rangle\,m$. The *set of all reachable markings* of N is denoted by $[m_0\rangle$.

An EN system is 1-live if every event can fire in, at least, one reachable marking.

Some properties of a net can be studied through the incidence matrix and its invariants. An S-invariant associates weights to conditions so that the weighted sum of tokens is the same in all reachable markings.

Definition 2. *Let N be a net and let M be its incidence matrix. A vector $\mathbf{I} : B \to \mathbb{N}$ is an S-invariant iff it is a solution of: $\mathbf{I}^T \circ M = \mathbf{0}$.*

Similarly, a T-invariant is defined as a vector $\mathbf{J} : E \to \mathbb{N}$ iff it is a solution of: $M \circ \mathbf{J} = \mathbf{0}$.

An S-invariant is *monomarked* iff its coefficients are in $\{0, 1\}$ and exactly one condition corresponding to a 1 in the invariant belongs to the initial marking m_0.

In the following, when we write N_i we will refer to an EN system: $N_i = (B_i, E_i, F_i, m_0^i)$.

Relations between EN systems can be expressed by N-morphisms ([7]), corresponding to a form of partial simulation. $\widehat{N}$-morphisms are a special case of N-morphisms and will be used in defining the operation of composition.

Definition 3. *A $\widehat{N}$-morphism from N_1 to N_2 is a pair (β, η), such that:*

1. *$\beta \subseteq B_1 \times B_2$, and $\beta^{-1} : B_2 \to B_1$ is a total and injective function;*
2. *$\eta : E_1 \to^* E_2$ is a partial and surjective function;*
3. *if $\eta(e_1)$ is undefined, then $\beta({}^\bullet e_1{}^\bullet) = \emptyset$;*
4. *if $\eta(e_1) = e_2$, then $\beta({}^\bullet e_1) = {}^\bullet e_2$ and $\beta(e_1{}^\bullet) = e_2{}^\bullet$;*
5. *$\forall (b_1, b_2) \in \beta : [b_1 \in m_0^1 \Leftrightarrow b_2 \in m_0^2]$.*

$\widehat{N}$-morphisms reflect S-invariants ([1]), but do not preserve them.

We recall an operation of composition (defined in [8]) that composes two EN systems, N_1 and N_2, with respect to a third EN system N_I called interface because it expose the protocol of interaction between the two systems. The composition is driven by a pair of $\widehat{N}$-morphisms, (β_1, η_1) and (β_2, η_2), respectively from N_1 to N_I, and from N_2 to N_I. In this way, N_1 and N_2 can be seen as composed each one by a local component and a component isomorphic to N_I.

Definition 4. *Let $D_i = \{b \in B_i | \beta_i(b) \neq \emptyset\}$, and $G_i = dom(\eta_i)$.*

We define $N_1\langle N_I\rangle N_2 = N = (B, E, F, m_0)$ as follows:

1. $B = (B_1 \setminus D_1) \cup (B_2 \setminus D_2) \cup B_I$;
2. $E = (E_1 \setminus G_1) \cup (E_2 \setminus G_2) \cup E_{sync}$,
   where $E_{sync} = \{\langle e_1, e_2\rangle | e_1 \in G_1, e_2 \in G_2, \eta_1(e_1) = \eta_2(e_2)\}$;
3. F *is defined by the following clauses:*
   (a) $\forall b \in (B_i \setminus D_i), \forall e \in (E_i \setminus G_i), i = 1, 2$ *we have* $(b, e) \in F \Leftrightarrow (b, e) \in F_i$ *and* $(e, b) \in F \Leftrightarrow (e, b) \in F_i$;
   (b) $\forall b \in (B_i \setminus D_i), \forall e \in G_i, \forall e_j \in E_{3-i}$ *and* $e_s = \langle e, e_j\rangle$ *if* $i = 1$ *or* $e_s = \langle e_j, e\rangle$ *if* $i = 2$, *we have* $(b, e_s) \in F \Leftrightarrow e_s \in E, (b, e) \in F_i$ *and* $(e_s, b) \in F \Leftrightarrow e_s \in E, (e, b) \in F_i$;
   (c) $\forall b \in B_I, \forall e = \langle e_1, e_2\rangle \in E_{sync}$ *we have* $(b, e) \in F \Leftrightarrow (\beta_1^{-1}(b), e_1) \in F_1, (\beta_2^{-1}(b), e_2) \in F_2$ *and* $(e, b) \in F \Leftrightarrow (e_1, \beta_1^{-1}(b)) \in F_1, (e_2, \beta_2^{-1}(b)) \in F_2$;
4. $m_0 = (m_0^1 \setminus D_1) \cup (m_0^2 \setminus D_2) \cup m_0^I$.

From this construction it follows immediately that $N = N_1\langle N_I\rangle N_2$ *as above is an EN system.*

The pair (γ_i, δ_i), *with* $\gamma_i \subseteq B \times B_i$ *and* $\delta_i : E \to E_i$ *defined as:*

- $\gamma_i = \{(b, b) | b \in B_i \setminus D_i\} \cup \{(b, \beta_i^{-1}(b)) | b \in B_I\}$,
- $\forall e \in E_i \setminus G_i : \delta_i(e) = e, \delta_{3-i}(e) =$ *undefined,*
- $\forall \langle e_1, e_2\rangle \in E_{sync} : \delta_i(\langle e_1, e_2\rangle) = e_i$.

is an $\widehat{N}$*-morphism from* $N = N_1\langle N_I\rangle N_2$ *to* $N_i, i = 1, 2$.

Informally, the composition creates a new EN system with the original conditions, events and arcs local to the components plus the conditions of the interface and the Cartesian product of the events to be synchronized. Synchronized events are connected to the local conditions, if there is an arc in the components between these objects, and to the conditions of the interface, if there is an arc in both the components between these events and the inverse-image of the conditions of the interface.

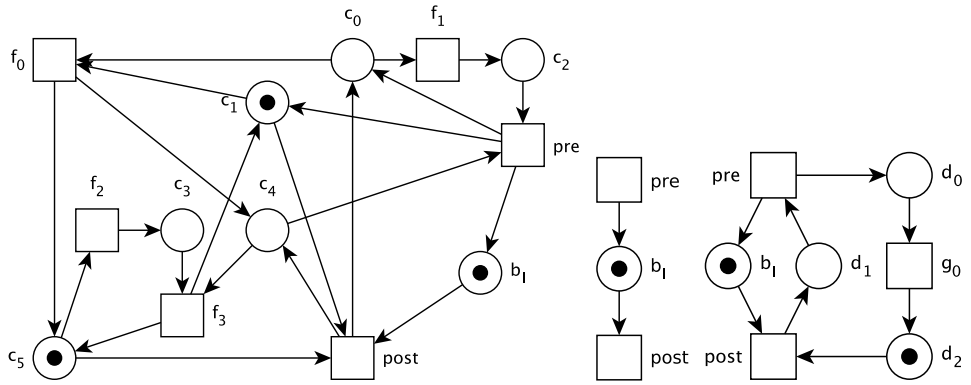
In Fig. 1 it is shown an example of the two EN systems to be composed and the interface; in Fig. 2 there is the resulting net. The $\widehat{N}$-morphisms are defined by identical labels on conditions and events.

Composition through $\widehat{N}$-morphisms assure that, if a component N_1 is bisimilar to the interface, then the composed net is bisimilar to the other component, N_2 [1].

Bisimulation relations have been introduced as an equivalence notion with respect to event observation [5]. We define the observability of events of a system by using a labelling function which associates the same label to different events, when viewed as equal by an observer, and the label τ to unobservable events.

Definition 5. *Let* $N = (B, E, F, m_0)$ *be an Elementary Net System,* $l : E \to L \cup \{\tau\}$ *be a labelling function where* L *is the alphabet of observable actions and* $\tau \notin L$ *the unobservable action. Let* ϵ *denote the empty word in both* E^* *and* L^*. *The function* l *is extended to a homomorphism* $l : E^* \to L^*$ *in the following way:*

$$l(\epsilon) = \epsilon$$



(a) N_1 (b) N_I (c) N_2

Fig. 1: The EN systems N_1 and N_2 being composed through the interface N_I

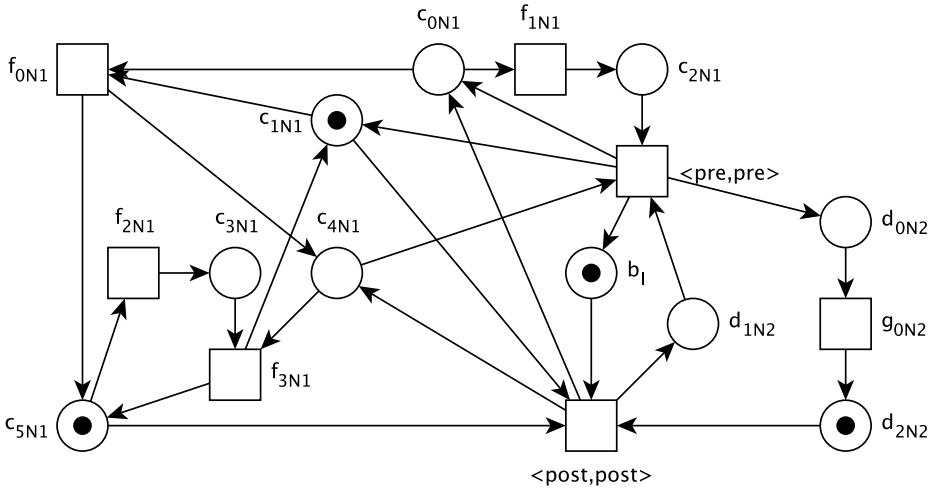


Fig. 2: The resulting EN system $N_1\langle N_I\rangle N_2$

$$\forall e \in E, \forall w \in E^*, l(ew) = \begin{cases} l(e)l(w) & \text{if } l(e) \neq \tau \\ l(w) & \text{if } l(e) = \tau \end{cases}$$

The pair (N, l) *is called* Labelled *Elementary Net System.*

Let $m, m' \in [m_0\rangle$ *and* $a \in L \cup \{\epsilon\}$ *then:*

- a *is enabled at* m*, denoted* $m\,(a\rangle$*, iff* $\exists w \in E^* : l(w) = a$ *and* $m\,[w\rangle$*;*
- *if* a *is enabled at* m*, then the occurrence of* a *can lead from* m *to* m'*, denoted* $m\,(a\rangle\,m'$*, iff* $\exists w \in E^* : l(w) = a$ *and* $m\,[w\rangle\,m'$*.*

We define weak bisimulation as a relation between reachable markings of Labelled Elementary Net Systems [9].

Definition 6. *Let* $N_i = (B_i, E_i, F_i, m_0^i)$ *be an Elementary Net System for* $i = 1, 2$*, with the labelling function* $l_i : E_i \to L \cup \{\tau\}$*. Then* (N_1, l_1) *and* (N_2, l_2) *are* weakly bisimilar*, denoted* $(N_1, l_1) \approx (N_2, l_2)$*, iff* $\exists r \subseteq [m_0^1\rangle \times [m_0^2\rangle$ *such that:*

- $(m_0^1, m_0^2) \in r$*;*
- $\forall (m_1, m_2) \in r, \forall a \in L \cup \{\epsilon\}$ *it holds*

$$\forall m_1' : m_1\,(a\rangle\,m_1' \Rightarrow \exists m_2' : m_2\,(a\rangle\,m_2' \wedge (m_1', m_2') \in r$$

and (vice versa)

$$\forall m_2' : m_2\,(a\rangle\,m_2' \Rightarrow \exists m_1' : m_1\,(a\rangle\,m_1' \wedge (m_1', m_2') \in r$$

Such a relation r *is called* weak bisimulation*.*

As example, consider the systems N_2 and N_I of Fig. 1. The observable actions are the ones on E_I. As labelling function for N_2 take l_2 that maps each event on the correspondent one in E_I but for g_0 that is mapped on τ. As labelling function for N_I take the identity function. Now we can write $\{b_I, d_2\}\,(post\rangle\,\{d_1\}$ because we have $\{g_0, post\} \in E_2^*$ such that $l_2(\{g_0, post\}) = post$ and $\{b_I, d_2\}\,[\{g_0, post\}\rangle\,\{d_1\}$.

For simplicity, in the remaining part of the paper we will use the term *bisimulation* instead of *weak bisimulation.*

3 Visibility

Let us consider two EN systems, the defender N_D and the attacker N_A, together with their composition on the interface N_I: $N_D\langle N_I\rangle N_A$ as defined above.

In the following definitions, we will use invariants and markings either as vectors or as characteristic functions: if $\mathbf{v}$ is a vector $x \in \mathbf{v} \Leftrightarrow \mathbf{v}(x) \neq 0$. Since the whole system can be seen as composition of subsystems, we can restrict every vector to the components belonging to a given subsystem. We will use the symbol $\downarrow$ for such a restriction. If $\mathbf{v}$ is a vector related to N, we can divide it in parts associated to the defender, the interface and the attacker: $\mathbf{v}_{\downarrow D}$, $\mathbf{v}_{\downarrow I}$, $\mathbf{v}_{\downarrow I \cup A}$ and $\mathbf{v}_{\downarrow A}$.

We can now define the observability that the attacker has on the markings of the whole system.

Definition 7. *The* attacker-view *of a marking m of the system N is the restriction of the marking on the conditions of N_A and N_I.*

$$\forall m \in [m_0\rangle\,, m_{\downarrow_{I\cup A}} = m \cap (B_A \cup B_I)$$

In general, the attacker is able to distinguish only subsets of markings of the composed system.

Definition 8. *We say that two distinct markings $m, m' \in [m_0\rangle$ are* attacker-view equivalent *if $m_{\downarrow_{I\cup A}} = m'_{\downarrow_{I\cup A}}$.*

A marking $m \in [m_0\rangle$ is distinguishable *by the attacker if $\neg\,\exists\, m' \in [m_0\rangle$: $m_{\downarrow_{I\cup A}} = m'_{\downarrow_{I\cup A}}$.*

The attacker has a complete distinguishability *of the markings of the whole system if:*

$$\forall m, m' \in [m_0\rangle\,, m_{\downarrow_{I\cup A}} = m'_{\downarrow_{I\cup A}} \Rightarrow m = m'$$

The interesting cases are the ones in which there is no complete distinguishability. We define as follows the conditions visible or invisible to the attacker.

Definition 9. *Condition $p \in B_D \setminus B_I$ is* invisible from a marking $m_A \in \left[m_0^A\right\rangle$ *for an attacker N_A, in isolation, iff*

$$\exists\, m, m' \in [m_0\rangle : m(p) = 0 \wedge m'(p) = 1 \wedge m_{\downarrow_{I\cup A}} = m'_{\downarrow_{I\cup A}} = m_A$$

Condition $p \in B_D \setminus B_I$ is invisible *for N_A iff p is invisible for every $m_A \in \left[m_0^A\right\rangle$. If a condition is not invisible then we will say that is visible.*

We will call $S_D \subseteq B_D \setminus B_I$ the set of invisible conditions computed as in the procedure reported below for *an* attacker N_A, such that N_A is composed with N_D through the interface N_I.

We will call $S_D^* \subseteq B_D \setminus B_I$ the set of invisible conditions for *all* attacking net systems N_A, such that N_A is composed with N_D through the interface N_I.

3.1 Invisible and visible conditions: results

To determine which conditions are in S_D we have to follow this procedure:

- partition the reachable markings of the composed system according to the markings of the attacker;
- for each marking of the attacker, compute the invisible conditions and
- compute the intersection of the sets of invisible conditions above.

Since the computation of all the markings of a Petri Net is exponential, to find the set of invisible conditions is an exponential computation too.

Let us explain this procedure by means of the example of Fig. 1. We use the markings of the composed system, showed in Table 1, and of the attacker, Table 2, to compute S_D. Starting by the markings of the attacker N_2, let us partition the markings of the composed system in sets of undistinguishable markings as

	b_I	c_{0N1}	c_{1N1}	c_{2N1}	c_{3N1}	c_{4N1}	c_{5N1}	d_{0N2}	d_{1N2}	d_{2N2}
S_0	1	0	1	0	0	0	1	0	0	1
S_1	0	1	0	0	0	1	0	0	1	0
S_2	1	0	1	0	1	0	0	0	0	1
S_3	0	0	0	1	0	1	0	0	1	0
S_4	1	1	1	0	0	0	0	1	0	0
S_5	1	1	1	0	0	0	0	0	0	1
S_6	1	0	1	1	0	0	0	1	0	0
S_7	1	0	0	0	0	1	1	1	0	0
S_8	1	0	1	1	0	0	0	0	0	1
S_9	1	0	0	0	0	1	1	0	0	1
S_{10}	1	0	0	0	1	1	0	1	0	0
S_{11}	1	0	0	0	1	1	0	0	0	1
S_{12}	1	0	1	0	0	0	1	1	0	0
S_{13}	1	0	1	0	1	0	0	1	0	0

Table 1: Reachable states of system $N_1\langle N_I\rangle N_2$ of Fig. 2

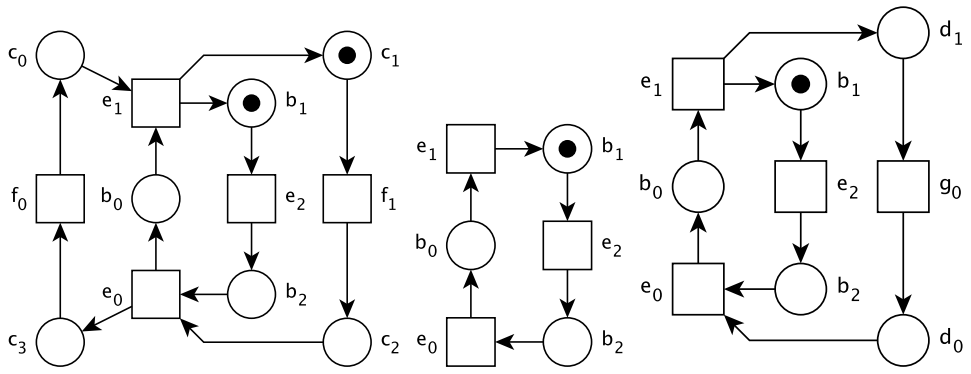
	b_I	d_0	d_1	d_2	possible markings of the composed system	conditions invisible
S_{0A}	1	0	0	1	S_0, S_2, S_5, S_8, S_9, S_{11}	$\{c_{0N1}, c_{1N1}, c_{2N1}, c_{3N1}, c_{4N1}, c_{5N1}\}$
S_{1A}	0	0	1	0	S_1, S_3	$\{c_{0N1}, c_{2N1}\}$
S_{2A}	1	1	0	0	S_4, S_6, S_7, S_{10}, S_{12}, S_{13}	$\{c_{0N1}, c_{1N1}, c_{2N1}, c_{3N1}, c_{4N1}, c_{5N1}\}$

Table 2: Reachable states of system N_2 of Fig. 1c

in Table 2. In the same table are as well listed the conditions invisible from each marking of the attacker; the conditions invisible for N_2 are $\{c_{0N1}, c_{2N1}\}$ given by the intersection of all of the computed S_D sets.

In order to compute S_D^*, we should construct every possible attacker compatible with the interface N_I in respect to the composition operation. This is obviously impossible and we cannot compute the set of conditions invisible to every attacker. Nevertheless, we conjecture that the conditions invisible to the interface (or to an attacker isomorphic to the interface) allow to infer a limit to the set S_D^*. The cases in which the attacker is bisimilar to the interface are discussed below.

Note that we are not interested in *controlling* the behaviour of the defender by imposing a specific marking of the attacker. This situation, at the extreme consequences, could be seen as a deadlock situation imposed by an attacker that blocks completely the interface. Consequently, we are not interested in, for example, a visible condition that is constant in every marking of the composed system since this would be a situation of (local) deadlock related to an attacker taking explicit control of the the defender by but not to the concept of visibility.



(a) N_D (b) N_I (c) N_A

Fig. 3: Two EN systems to be composed through the interface N_I

Let us now prove the central result. We define a necessary constraint for a defined attacker N_A such that a condition of the defender is not in S_D. This situation happens when a condition of the defender is in a monomarked invariant with a condition of the interface. In this case, it is possible to construct an attacker (isomorphic to the interface itself) with a marking in which that condition is visible.

Theorem 1. *Let N_D, N_I be bisimilar EN systems, and $(\beta_D, \eta_D) : N_D \to N_I$ an $\widehat{N}$-morphism. If N_I is 1-live and $b \in B_D \setminus \beta_D^{-1}(B_I), i \in \beta_D^{-1}(B_I)$ satisfies*

$b, i \in I_D$ *with* I_D *monomarked S-invariant of* N_D*, then* b *is visible for each attacker bisimilar to the interface.*

Proof. Consider an attacker isomorphic to the interface, $N_A = N_I$. Given that we consider each attacker bisimilar to the interface, if we prove that this result hold for the interface, it holds for all these attackers too.

Since S-invariants are reflected, I_D is an invariant of the composed net (that in this case is isomorphic to N_D). So, if we can reach a marking in which $i = 1$ then we are sure that $b = 0$ and then b is visible. If $m_0(i) = 1$ this is the marking we are looking for. Suppose $m_0(i) = 0$. Since N_I is an EN system, $\beta_D(i)$ is not isolated. If ${}^\bullet\beta_D(i) = \emptyset$, then $\beta_D(i)$ should have at least a post-event. In this case this post-event is dead while N_I is 1-live by hypothesis. So, the preset of $\beta_D(i)$ is not empty. Given that N_I is 1-live, an event in the preset of $\beta_D(i)$ will fire at some reachable case. Let us call $u \in E_I^*$ a sequence of events such that $m_0^I [u\rangle m_1^I$ and $m_1^I(\beta_D(i)) = 1$. From the assumption that $N_D \approx N_I$ with the labelling function $h : E_D \to E_I \cup \{\tau\}$ we can deduce that $\exists w \in E_D^* : h(w) = u, m_0^D [w\rangle m_1^D, m_1^D(i) = 1$. □

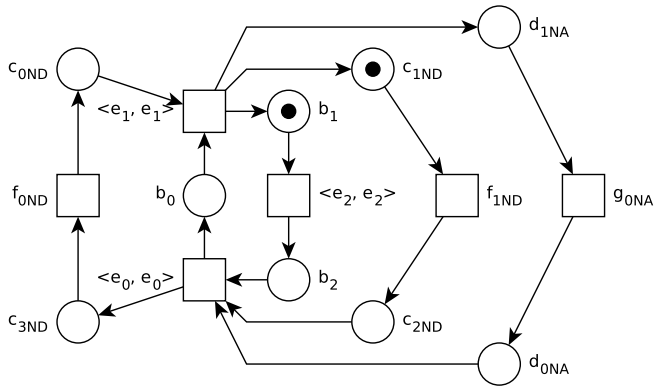


Fig. 4: The composition of the EN systems of Fig. 3

Note that taking into account an attacker not bisimilar to the interface is not of interest because this attacker can introduce some restrictions of behaviour, hence hiding some visible part of the defender. We can see an example of this case in Fig. 3 where the $\widehat{N}$-morphisms are implicitly defined by the identical labels of conditions and events.

If we modify the initial marking m_0 by adding a token in condition d_1 of net N_A, the attacker becomes bisimilar to the interface. In this case, conditions c_1, c_2, c_3 and c_4 of N_D are visible. If we consider the net system as it is, c_1 and c_2 are not visible, as we can see in Fig. 4 and Tables 3 and 4.

Asking a defender bisimilar to the interface is reasonable, because the interface is the protocol of interaction exposed by the defender, so we expect that

	b_0	b_1	b_2	c_{0ND}	c_{1ND}	c_{2ND}	c_{3ND}	d_{0NA}	d_{1NA}
S_0	0	1	0	0	1	0	0	0	0
S_1	0	0	1	0	1	0	0	0	0
S_2	0	1	0	0	0	1	0	0	0
S_3	0	0	1	0	0	1	0	0	0

Table 3: Reachable states of system $N_D\langle N_I\rangle N_A$ of Fig. 4

	b_0	b_1	b_2	d_0	d_1	possible markings of the composed system	invisible conditions
S_{0A}	0	1	0	0	0	S_0, S_2	$\{c_{1ND}, c_{2ND}\}$
S_{1A}	0	0	1	0	0	S_1, S_3	$\{c_{1ND}, c_{2ND}\}$

Table 4: Reachable states of system N_A of Fig. 3c

the system respect his own contract. Also the constraint on the liveness of the interface is reasonable. The only constraint that is not so easy to respect is the one on the S-invariant, because compute all the invariants of an Elementary Net is exponential. Nevertheless, a lot of tools compute the invariant for a given net.

3.2 Measuring visibility

We can give a measure of the uncertainty related to visibility. Intuitively, visible or invisible conditions are opposite ends of some kind of *spectrum* of visibility and, in Def. 9, we do not weight the relative persistence of the invisible condition p in marking m or m'.

For example, in Table 2, attacker case S_{0A}, condition b_{0N1} is more frequently un-marked than marked. Consequently, we could consider b_{0N1} as a random variable whose average information content — persistence in a given local state — depends on the chosen marking of the attacker.

Traditionally, entropy is a measure of the uncertainty associated with a random variable. Consequently, a measure of the uncertainty of the marking for a given defender condition in a given attacker marking can be given, as usual in information science, by using Shannon's entropy:

the entropy H of a discrete random variable $X = \{x_1, ..., x_n\}$ with p denoting the probability mass function of X is $H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$.

Obviously, when $H(X) = 1$ condition X seen as random variable is totally invisible on the attacker marking considered while when $H(X) = 0$ it is visible.

For example, with reference to Table 2, let us calculate the entropy of b_{0N1} seen as variable with possible values in $\{0, 1\}$ with respect to the attacker marking S_{0A}. Marking S_{0A} "covers" $\{S_0, S_2, S_5, S_8, S_9, S_{11}\}$ and, with reference to Table 1, we can divide this set in two subsets: one in which $b_{0N1} = 0$, $\{S_0, S_2, S_8, S_9, S_{11}\}$, and one with $b_{0N1} = 1$, $\{S_5\}$. By plain computation of the relative frequencies of persistence in a state, the entropy is $H(b_{0N1}) =$

$-\sum_{i=1}^{2} p(x_i) \log_2 p(x_i) = -5/6 \log_2 5/6 - 1/6 \log_2 1/6 = 0,65$. So b_{0N1} in S_{0A} is invisible at 65%.

4 Conclusion

We aimed at defining structurally the notion of *visibility* between composed subsystems in order to isolate the unwanted information flows between an hypothetical *defender* system and an *attacker* system whose interactions are coordinated by an *interface*. The composition of these three subsystems is formally defined in terms of morphisms. In the context of information science, our work is naturally placed in the field of *non-interference* as reported in the introduction.

We managed to use traditional tools in the study of Petri Nets like *invariants*, for the definition of the properties of our interest. In the context of this work we did not use T-invariants because their are more related to the concept of controlling the defender than to the concept of visibility. Unfortunately we failed in having a full structural description since, for proving theorem 1, we had to make an hypothesis of *bisimulation* between the *defender* and the *interface*. Nevertheless, we reached a preliminary result in a direction worth to be explored further. Next steps will be in the direction of a finer characterization of the statistical dependency between the subsystems, in proving the conjecture concerning the dependence between all the possible *attackers* and the *interface*, and in using different kinds of morphisms for the definition of the composition in order to avoid the use of bisimilarity relations in the proofs.

Acknowledgments Work partially supported by MIUR.

References

1. Luca Bernardinello, Elena Monticelli, and Lucia Pomello. On preserving structural and behavioural properties by composing net systems on interfaces. *Fundam. Inform.*, 80(1-3):31–47, 2007.
2. Eike Best, Philippe Darondeau, and Roberto Gorrieri. On the decidability of non interference over unbounded petri nets. In Konstantinos Chatzikokolakis and Véronique Cortier, editors, *SecCo*, volume 51 of *EPTCS*, pages 16–33, 2010.
3. Nadia Busi and Roberto Gorrieri. A survey on non-interference with petri nets. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 328–344. Springer, 2003.
4. Riccardo Focardi and Roberto Gorrieri. Classification of security properties (part I: Information flow). In Riccardo Focardi and Roberto Gorrieri, editors, *FOSAD*, volume 2171 of *Lecture Notes in Computer Science*, pages 331–396. Springer, 2000.
5. Robin Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.
6. Edward F. Moore. Gedanken-experiments on sequential machines. In Claude Elwood Shannon and John McCarthy, editors, *Automata Studies*, volume 34 of *Annals of mathematics studies*, pages 129–153. Princeton University Press, 1956.

7. Mogens Nielsen, Grzegorz Rozenberg, and P. S. Thiagarajan. Elementary transition systems. *Theor. Comput. Sci.*, 96(1):3–33, 1992.
8. Lucia Pomello and Luca Bernardinello. Formal tools for modular system development. In J. Cortadella and W. Reisig, editors, *ICATPN*, volume 3099 of *Lecture Notes in Computer Science*, pages 77–96. Springer, 2004.
9. Lucia Pomello, Grzegorz Rozenberg, and Carla Simone. A survey of equivalence notions for net based systems. In Grzegorz Rozenberg, editor, *Advances in Petri Nets: The DEMON Project*, volume 609 of *Lecture Notes in Computer Science*, pages 410–472. Springer, 1992.
10. Grzegorz Rozenberg and Joost Engelfriet. Elementary net systems. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 12–121. Springer, 1996.

Preface

This volume contains the papers presented at CompoNet 2012: second international workshop on Petri nets Compositions (and other models of concurrency) held on June 26th 2012 in Hamburg, Germany as a part of the International Conference on Applications and Theory of Petri Nets (PETRI NETS 2012).

This workshop aims at offering to researchers, using or developing compositions within their specific Petri net variants or related models of concurrency, a forum promoting cross-discussion and cross-fertilisation with the hope to enable the emergence of novel models of Petri nets compositions, dedicated to various application domains.

We would like to thank the authors of submitted papers for their interest in CompoNet. We also thank the program committee members and the external reviewers for their efficient work during the reviewing process. Last but not least, we thank the authors of the EasyChair conference management system which made the practical organisation of the reviewing process considerably easier.

June 2012

Hanna Klaudel and Franck Pommereau (PC chairs)

Program committee

External reviewers

Composition of Elementary Net Systems based on α-morphisms

Luca Bernardinello, Elisabetta Mangioni, and Lucia Pomello

Dipartimento di Informatica, Sistemistica e Comunicazione,
DISCo - Università degli Studi di Milano-Bicocca
viale Sarca, 336, I-20126 Milano, Italia
mangioni@disco.unimib.it

Abstract. In the development of distributed systems a central role is played by formal tools supporting various aspects of modularity such as compositionality, refinement and abstraction. One of the main challenges consists in developing methods allowing to derive properties of the composed system from properties of the components. In this context we consider Elementary Net Systems related by morphisms and compose them through an interface. Imposing structural constraints on the components, we obtain some structural properties of the composed system and, requiring additional local behavioural constaints, behavioural properties.

Keywords: Elementary Net Systems, morphisms, composition

1 Introduction

In the development of distributed systems a central role is played by formal tools supporting various aspects of modularity such as compositionality, refinement and abstraction. Several formal approaches are available. One of the main challenges consists in developing languages and methods allowing to derive properties of the refined or composed system from properties of the components.

In this paper we present a composition operator such that, by imposing on the components structural constraints and only local behavioural constraints, the composed system inherits behavioural properties of the components.

We consider systems modelled by State Machine Decomposable Elementary Net Systems, i.e.: Elementary Net Systems obtained by composing state machines through synchronized events.

Following the approach proposed in [13, 1, 4, 5], the basic idea consists in composing two different refinements of a common abstract view, obtaining a new model which describes the system comprising the details of both operands, while respecting the same abstract view.

The rules for identifying elements of the nets being composed are expressed by means of morphisms towards another net system, called interface. The interface can be seen as an abstraction of the whole system, shared by the components or, alternatively, it can be interpreted as the specification of the communication

protocol with which the components agree. In this case, each operand can be seen as made of the actual, local, component, and of an interface to the rest of the system. Even if this operation is not a limit in the category of nets here considered, the composed system results to be related to both the components and the interface by means of morphisms, and the resulting diagram is commutative.

The use of products in a suitable category of nets as a way to model composition by synchronization has been studied by several authors. A variation on this theme, more similar to ours, proposed by Fabre in [8], applies to safe nets and is built on the notion of pullback.

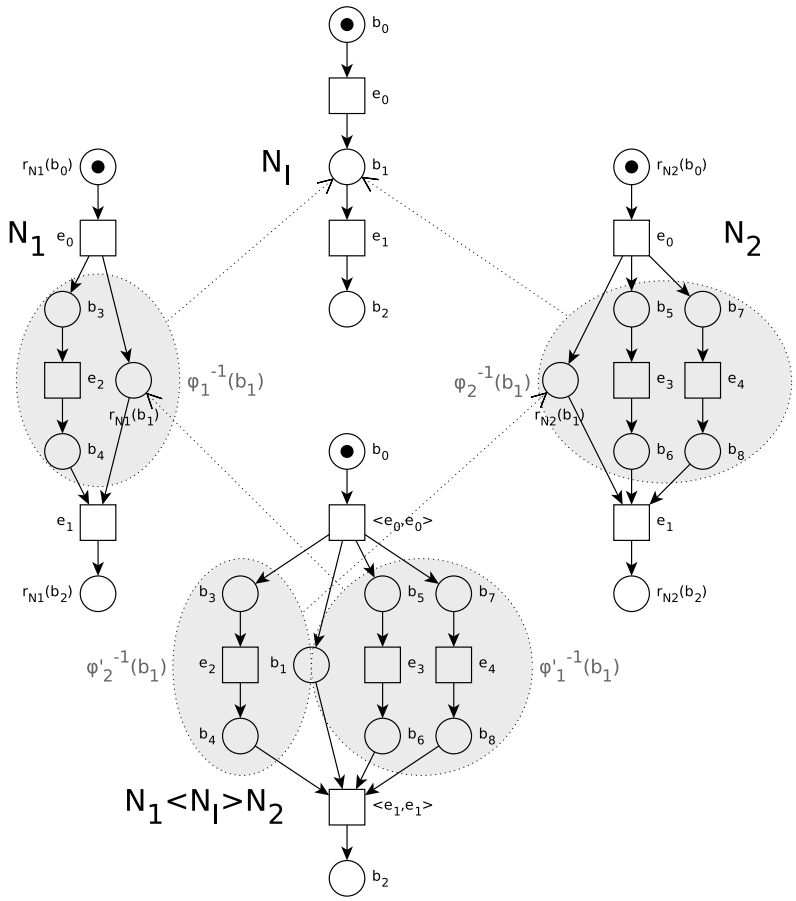


Fig. 1: An example of composition based on α-morphisms

Using morphisms to formalize the relation between a refined net and a more abstract one is not new. The majority of refinement approaches introduced in Petri net theory are mainly based on transition refinement and, less frequently,

on place refinement; see [9] and for a survey [6]. Another survey paper, [12], describes a set of techniques which allow to refine transitions in Place/transition nets, so that the relation between the abstract net and its refinement is given by a morphism. There, the emphasis is on refinement rules that preserve specific behavioural properties, within the wider context of general transformation rules on nets.

A very general class of morphisms, interpreted as abstraction of system requirements, with less focus on strict preservation of behavioural properties, is defined in [7].

The refinement we use in this paper is similar in spirit to the one proposed in [11]. In that approach, refinement is defined on transition systems, however it is strictly related to refinement of local states in nets, through the notion of region.

The morphisms used in this paper, called α-morphisms, can be seen as a special case of those introduced by Winskel in [17]. Other morphisms in the same line of Winskel morphisms, are the ones given in [16] and [2].

A simple example shows the main features of our proposal (see Fig. 1). The interface, N_I, is a simple sequence of two events. The two components, N_1 and N_2, refine the same local state, b_1, each by a subnet, shown on a gray background. The composed net, $N_1\langle N_I\rangle N_2$, contains both refinements of b_1, while the rest of the net, not refined by the components, is taken as it is.

The paper is structured as follows. In Section 2 we collect preliminary definitions related to Petri nets which are used in the rest of the paper. Section 3 contains the definition of α-morphisms and their properties. Section 4 contains the definition of $\widehat{N}$-morphisms [13] and their properties. Section 5 defines the composition guided by α-morphisms and the main result of the paper: under some structural and local behavioural properties the composed net is bisimilar to its components. Finally, in Section 6 we discuss some critical issues in our approach. Proofs omitted in this paper can be found in an extended version [3].

2 Preliminary definitions

In this section, we recall the basic definitions of net theory, in particular Elementary Net Systems [15].

We will use the symbol $\downarrow$ to denote the restriction of a function on a subset of its domain.

2.1 Petri Nets

In net theory, models of distributed systems are based on objects called nets which specify local states, local transitions and the relations among them. A *net* is a triple $N = (B, E, F)$, where B is a set of *conditions* or local states, E is a set of *events* or transitions such that $B \cap E = \emptyset$ and $F \subseteq (B \times E) \cup (E \times B)$ is the *flow relation*.

We adopt the usual graphical notation: conditions are represented by circles, events by boxes and the flow relation by arcs. The set of elements of a net will be denoted by $X = B \cup E$; note that we allow nets with isolated elements.

The *preset* of an element $x \in X$ is ${}^\bullet x = \{y \in X | (y, x) \in F\}$; the *postset* of x is $x^\bullet = \{y \in X | (x, y) \in F\}$; the *neighbourhood* of x is given by ${}^\bullet x^\bullet = {}^\bullet x \cup x^\bullet$. These notations are extended to subsets of elements in the usual way.

For any net N we denote the *in-elements* of N by ${}^\bigcirc N = \{x \in X_N : {}^\bullet x = \emptyset\}$ and the *out-elements* of N by $N^\bigcirc = \{x \in X_N : x^\bullet = \emptyset\}$.

A net is *simple* if for all $x, y \in X$, if ${}^\bullet x = {}^\bullet y$ and $x^\bullet = y^\bullet$, then $x = y$.

A net $N' = (B', E', F')$ is a *subnet* of $N = (B, E, F)$ if $B' \subseteq B, E' \subseteq E$, and $F' = F \cap ((B' \times E') \cup (E' \times B'))$. Given a subset of elements $A \subseteq X$, we say that $N(A)$ is the *subnet of N identified* by A if $N(A) = (B \cap A, E \cap A, F \cap (A \times A))$.

A *State Machine* is a connected net such that each event e has exactly one input condition and exactly one output condition: $\forall e \in E, |{}^\bullet e| = |e^\bullet| = 1$.

Elementary Net (EN) Systems are a basic system model in net theory. An *Elementary Net System* is a quadruple $N = (B, E, F, m_0)$, where (B, E, F) is a net such that B and E are finite sets, self-loops are not allowed, isolated elements are not allowed, and the *initial marking* is $m_0 \subseteq B$.

The elements in the initial marking are interpreted as the conditions which are true in the initial state.

A subnet of an Elementary Net System N identified by a subset of conditions A and all its pre and post events, $N(A \cup {}^\bullet A^\bullet)$, is a *Sequential Component* of N if $N(A \cup {}^\bullet A^\bullet)$ is a State Machine and if it has only one token in the initial marking.

An Elementary Net System is *covered* by Sequential Components if every condition of the net belongs to at least a Sequential Component. In this case we say that the system is *State Machine Decomposable*.

The behaviour of Elementary Net Systems is defined through the firing rule, which specifies when an event can occur, and how event occurrences modify the holding of conditions, i.e. the state of the system.

Let $N = (B, E, F, m_0)$ be an Elementary Net System, $e \in E$ and $m \subseteq B$. The event e is *enabled* at m, denoted $m\,[e\rangle$, if ${}^\bullet e \subseteq m$ and $e^\bullet \cap m = \emptyset$; the occurrence of e at m leads from m to m', denoted $m\,[e\rangle\, m'$, iff $m' = (m \setminus {}^\bullet e) \cup e^\bullet$.

Let ϵ denote the empty word in E^*. The firing rule is extended to sequences of events by $m\,[\epsilon\rangle\, m$ and $\forall e \in E, \forall w \in E^*, m\,[ew\rangle\, m' = m\,[e\rangle\, m''[w\rangle m'$; w is then called *firing sequence*.

A subset $m \subseteq B$ is a *reachable marking* of N if there exists a $w \in E^*$ such that $m_0\,[w\rangle\, m$. The *set of all reachable markings* of N is denoted by $[m_0\rangle$.

An Elementary Net System is *contact-free* if $\forall e \in E, \forall m \in [m_0\rangle$: ${}^\bullet e \subseteq m$ implies $e^\bullet \cap m = \emptyset$. If an Elementary Net System is covered by Sequential Components then it is contact-free. An event is called *dead* at a marking m if it is not enabled at any marking reachable from m. A reachable marking m is called *dead* if no event is enabled at m. An Elementary Net System is *deadlock-free* if no reachable marking is dead.

2.2 Unfoldings

The semantics of an Elementary Net System can be given as its *unfolding*. The unfolding is an acyclic net, possibly infinite, which records the occurrences of its elements in all possible executions.

Definition 1. *Let $N = (B, E, F)$ be a net, and let $x, y \in X$. We say that x and y are in* conflict*, denoted by $x \#_N y$, if there exist two distinct events $e_x, e_y \in E$ such that $e_x F^* x$, $e_y F^* y$, and ${}^\bullet e_x \cap {}^\bullet e_y \neq \emptyset$.*

Definition 2. *An occurrence net is a net $N = (B, E, F)$ satisfying:*

1. *if $e_1, e_2 \in E, e_1{}^\bullet \cap e_2{}^\bullet \neq \emptyset$ then $e_1 = e_2$;*
2. *F^* is a partial order,*
3. *for any $x \in X, \{y : yF^*x\}$ is finite;*
4. *$\#_N$ is irreflexive,*
5. *the minimal elements with respect to F^* are conditions.*

A branching process of N is an occurrence net whose elements can be mapped to the elements of N.

Definition 3. *Let $N = (B, E, F, m_0)$ be an Elementary Net System, and $\Sigma = (P, T, G)$ be an occurrence net. Let $\pi : P \cup T \to B \cup E$ be a map.*

The pair (Σ, π) is a branching process of N if:

- *$\pi(P) \subseteq B$, $\pi(T) \subseteq E$;*
- *π restricted to the minimal elements of Σ is a bijection on m_0;*
- *for each $t \in T$, π restricted to ${}^\bullet t$ is injective and π restricted to $t^\bullet$ is injective;*
- *for each $t \in T$, $\pi({}^\bullet t) = {}^\bullet(\pi(t))$ and $\pi(t^\bullet) = ({}^\bullet\pi(t))$.*

The unfolding of an Elementary Net System N, denoted by $Unf(N)$, is the "maximal" branching process of N, namely the unique branching process such that any other branching process of N is isomorphic to a subnet of $Unf(N)$. The map associated to the unfolding will be denoted u and called *folding*.

3 A class of morphisms

In the rest of the paper, we consider the class of State Machine Decomposable Elementary Net Systems (SMD-EN Systems).

In this section we give the formal definition of α-morphisms for this class of systems, and present some of their properties, particularly with respect to the preservation of both structural and behavioural properties, as formally introduced in [4].

We start by giving the formal definition of a general morphism and then present the more specific restrictions.

Definition 4. *Let $N_i = (B_i, E_i, F_i, m_0^i)$ be a SMD-EN System, for $i = 1, 2$. An ω-morphism from N_1 to N_2 is a total surjective map $\varphi : X_1 \to X_2$ such that:*

1. $\varphi(B_1) = B_2$;
2. $\varphi(m_0^1) = m_0^2$;
3. $\forall e_1 \in E_1$, *if* $\varphi(e_1) \in E_2$, *then* $\varphi({}^\bullet e_1) = {}^\bullet\varphi(e_1)$ *and* $\varphi(e_1{}^\bullet) = \varphi(e_1)^\bullet$;
4. $\forall e_1 \in E_1$, *if* $\varphi(e_1) \in B_2$, *then* $\varphi({}^\bullet e_1{}^\bullet) = \{\varphi(e_1)\}$;

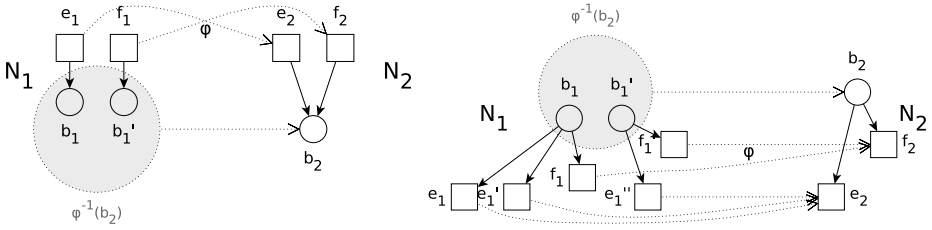
We require that the map is total and surjective because N_1 refines the abstract model N_2, and any abstract element must be related to its refinement.

In particular, a subset of nodes can be mapped on a single condition $b_2 \in B_2$, in this case, we will call *bubble* the subnet identified by this subset $N_1(\varphi^{-1}(b_2))$; if more than one element is mapped on b_2, we will say that b_2 is *refined* by φ. As example, we can see in Fig. 1 the refinement of condition b_1 of N_I with the bubble enclosed in the shaded oval on N_1.

The additional constraints listed in the next definition will be explained below through simple examples.

Definition 5. *Let* $N_i = (B_i, E_i, F_i, m_0^i)$ *be a SMD-EN System, for* $i = 1, 2$. *An* α*-morphism from* N_1 *to* N_2 *is an* ω*-morphism with the following additional constraints:*

5. $\forall b_2 \in B_2$:
   (a) $N_1(\varphi^{-1}(b_2))$ *is an acyclic net;*
   (b) $\forall b_1 \in {}^\bigcirc N_1(\varphi^{-1}(b_2)),\ \varphi({}^\bullet b_1) \subseteq {}^\bullet b_2$ *and* $({}^\bullet b_2 \neq \emptyset \Rightarrow {}^\bullet b_1 \neq \emptyset)$;
   (c) $\forall b_1 \in N_1(\varphi^{-1}(b_2))^\bigcirc,\ \varphi(b_1{}^\bullet) = b_2{}^\bullet$;
   (d) $\forall b_1 \in \varphi^{-1}(b_2) \cap B_1$,
   $(b_1 \notin {}^\bigcirc N_1(\varphi^{-1}(b_2)) \Rightarrow \varphi({}^\bullet b_1) = \{b_2\})$ *and* $(b_1 \notin N_1(\varphi^{-1}(b_2))^\bigcirc \Rightarrow \varphi(b_1{}^\bullet) = \{b_2\})$;
   (e) $\forall b_1 \in \varphi^{-1}(b_2) \cap B_1$, *there is a Sequential Component* N_{SC} *of* N_1 *such that* $b_1 \in B_{SC}$ *and* $\varphi^{-1}({}^\bullet b_2{}^\bullet) \subseteq E_{SC}$.



(a) Pre events of an in-condition (b) Post events of an out-condition

Fig. 2: Pre and post event of a bubble

As we can see in Fig. 2a and 2b, in-conditions and out-conditions have different constraints, 5b and 5c respectively. As required by 5c, we do not allow that choices, which are internal to a bubble, constrain a final marking of that bubble: i.e., each out-condition of the bubble must have the same choices of the

condition it refines. Instead, pre-events do not need this strict constraint (5b): hence it is sufficient that pre-events of any in-condition are mapped on a subset of the pre-events of the condition it refines. For example, in this particular case, we know that the choice between e_1 and f_1 of Figure 2a is made before the bubble, and this is implied also by the requirement 5e) on Sequential Components. Moreover, the conditions that are internal to a bubble must have pre-events and post-events which are all mapped to the refined condition b_2, as required by 5d.

By 5e, events in the neighbourhood of a bubble, as well as their images, can not be concurrent. However, within a bubble there can be concurrent events. By the combined effect of 5a-5e, in any execution, when a post-event of a bubble fires, in the next marking no local state within the bubble will be marked.

The α-morphisms are closed by composition, the identity function on X is an α-morphism, and the composition is associative. Hence, the family of SMD-EN Systems together with α-morphisms forms a category.

We now list some properties of α-morphisms which have been proved in [4]. Given an α-morphism $\varphi : N_1 \to N_2$ we can say that:

p1 the partition of the nodes of N_1 induced by φ can be lifted to a net structure: the class of nodes mapped to a place b becomes a place, while the class of nodes mapped to an event e becomes an event; the flow relation is defined in the obvious way. The resulting net is isomorphic to N_2;

p2 firing an output event of a bubble empties the bubble: Let $e_1 \in E_1, b_2 \in B_2$: $e_1 \in \varphi^{-1}(b_2{}^\bullet)$; $m_1, m_1' \in [m_0^1\rangle$: $m_1 \,[e_1\rangle\, m_1'$, then $m_1' \cap \varphi^{-1}(b_2) = \emptyset$;

p3 no input event of a bubble is enabled whenever a token is within the bubble: Let $e_1 \in E_1, b_2 \in B_2$: $e_1 \in \varphi^{-1}({}^\bullet b_2)$; $m_1, m_1' \in [m_0^1\rangle$: $m_1 \,[e_1\rangle\, m_1'$ then $m_1 \cap \varphi^{-1}(b_2) = \emptyset$;

p4 sequential components are reflected in the sense that the inverse image of a sequential component is covered by sequential components. Sequential components are not preserved;

p5 φ preserves reachable markings:
If $m_1 \in [m_0^1\rangle$ and $m_1 \,[e\rangle\, m_1'$ in N_1 then $\varphi(m_1) \in [m_0^2\rangle$ and
- if $\varphi(e) \in E_2$ then $\varphi(m_1) \,[\varphi(e)\rangle\, \varphi(m_1')$ else
- (if $\varphi(e) \in B_2$ then) $\varphi(m_1) = \varphi(m_1')$.

Stronger properties hold under additional constraints. Given an α-morphism $\varphi : N_1 \to N_2$, and a condition $b_2 \in B_2$ with its refinement $N_1(\varphi^{-1}(b_2))$, we define two new SMD-EN Systems. The first one, denoted $S_1(b_2)$, contains (a copy of) the subnet $N_1(\varphi^{-1}(b_2))$, its pre and post events in E_1 and two new conditions: b_1^{in}, which is pre of all the pre events, and b_1^{out}, which is post of all the post-events. The initial marking of $S_1(b_2)$ will be $\{b_1^{in}\}$ or, if there are no pre events, the initial marking of the bubble in N_1. The second system, denoted $S_2(b_2)$, contains b_2, its pre- and post-events and two new conditions: b_2^{in}, which is pre of all the pre-events, and b_2^{out}, which is post of all the post-events. The initial marking of $S_2(b_2)$ will be $\{b_2^{in}\}$ or, if there are no pre events, the initial marking of b_2. Define φ^S as a map from $S_1(b_2)$ to $S_2(b_2)$, which restricts φ to the elements of $S_1(b_2)$, and extends it with $\varphi^S(b_1^{in}) = b_2^{in}$ and $\varphi^S(b_1^{out}) = b_2^{out}$. Note that $S_1(b_2)$ and

$S_2(b_2)$ are SMD-EN Systems and that φ^S is an α-morphism. Let $Unf(S_1(b_2))$ be the unfolding of $S_1(b_2)$, with folding function $u : Unf(S_1(b_2)) \to S_1(b_2)$.

Consider the following additional constraints:

c1 the initial marking of each bubble is at the start of the bubble itself; formally: for each $b_2 \in B_2$ one of the following conditions hold
- $\varphi^{-1}(b_2) \cap m_0^1 = \emptyset$ or
- if ${}^\bullet b_2 \neq \emptyset$ then there is $e_1 \in \varphi^{-1}({}^\bullet b_2)$ such that $\varphi^{-1}(b_2) \cap m_0^1 = e_1{}^\bullet$ or
- if ${}^\bullet b_2 = \emptyset$ then $\varphi^{-1}(b_2) \cap m_0^1 = {}^\bigcirc\varphi^{-1}(b_2)$;

c2 any condition is refined by a subnet such that, when a final marking is reached, this one enables events which correspond to the post-events of the refined condition, i.e.:
$\varphi^S \circ u$ is an α-morphism from $Unf(S_1(b_2))$ to $S_2(b_2)$;

c3 different bubbles do not "interfere" with each other:
we say that two bubbles interfere with each other when their images share, at least, a neighbour.

The first condition assures that the initial marking of a bubble, if present, is in the initial conditions of the bubble and is generated by one of the pre-events, if there are some of them. The second condition is necessary to give to each final marking of a bubble the same choices that the abstract condition has. The third one is not restrictive since the refinement of two interfering conditions can be done in two different steps.

Under **c1**, **c2**, and **c3**, the following properties can be proved [4]:

p6 reachable markings of N_2 are reflected:
for all $m_2 \in [m_0^2\rangle$, there is $m_1 \in [m_0^1\rangle$ such that $\varphi(m_1) = m_2$;

p7 N_1 and N_2 are weakly bisimilar:
by using φ, define two labelling functions such that E_2 are all observable, i.e.: l_2 is the identity function, and the invisible events of N_1 are the ones mapped to conditions; then (N_1, l_1) and (N_2, l_2) are weakly bisimilar $(N_1, l_1) \approx (N_2, l_2)$.

For a definition of weak bisimulation of EN Systems see [14].

4 Relations with $\widehat{N}$-morphisms

The ω and α-morphisms here defined are related to $\widehat{N}$-morphisms, introduced in [13] and studied in [5], that are a restriction of N-morphisms defined in [10].

Here, we are interested in pointing out the precise relation, because we will apply to α-morphisms some results previously shown for $\widehat{N}$-morphisms.

First, let us recall the definition of $\widehat{N}$-morphisms.

Definition 6. *Let $N_i = (B_i, E_i, F_i, m_0)$ be an EN system for $i = 1, 2$.*

A $\widehat{N}$-morphism from N_1 to N_2 is a pair (β, η), where:

1. *$\beta \subseteq B_1 \times B_2$ and $\beta^{-1} : B_2 \to B_1$ is a total and injective function;*

2. $\eta : E_1 \rightarrow_* E_2$ *is a partial and surjective function;*
3. *if* $\eta(e_1)$ *is undefined, then* $\beta({}^\bullet e_1) = \emptyset = \beta(e_1{}^\bullet)$*;*
4. *if* $\eta(e_1) = e_2$*, then* $\beta({}^\bullet e_1) = {}^\bullet e_2$ *and* $\beta(e_1{}^\bullet) = e_2{}^\bullet$*;*
5. $\forall (b_1, b_2) \in \beta : [b_1 \in m_0^1 \Leftrightarrow b_2 \in m_0^2]$.

In order to compare ω- and α-morphisms with $\widehat{N}$-morphisms, we need some auxiliary notions. Given an ω-morphism φ from N_1 to N_2, we say that N_1 is *canonical* with respect to φ if, for each bubble induced by φ, it contains a local state corresponding to the image of the bubble.

Definition 7. *Let* $\varphi : X_1 \rightarrow X_2$ *be an* ω*-morphism from* N_1 *to* N_2*.* N_1 *is* canonical *with respect to* φ *if for each* $b_2 \in B_2$*, there exists a unique* $b_1 \in \varphi^{-1}(b_2) \cap B_1$ *satisfying:*

- $b_1 \in m_0^1 \Leftrightarrow b_2 \in m_0^2$*;*
- ${}^\bullet b_1 = \varphi^{-1}({}^\bullet b_2)$*;*
- $b_1{}^\bullet = \varphi^{-1}(b_2{}^\bullet)$.

In this case, b_1 *is said to be the* representation *of* b_2*, denoted* $r_{N_1}(b_2)$*. We define the subnet of a bubble, obtained by removing the representation:* $N_1^{-rep}(b_2) = N_1(\varphi^{-1}(b_2) \setminus \{r_{N_1}(b_2)\})$.

If N_1 is not canonical, it is always possible to construct its unique canonical version, $N_1^{\mathcal{C}}$, either by adding the missing representations (and marking them as their images) or by deleting multiple representations. The corresponding morphism, $\varphi^{\mathcal{C}}$, coincides with φ, plus the mapping of new conditions on the corresponding conditions of N_2. It is easy to verify that the canonical version of a system, with respect to an α-morphism to another SMD-EN System, is unique up to isomorphisms.

We have proved in [4] that $\varphi^{\mathcal{C}}$ is an ω-morphism from $N_1^{\mathcal{C}}$ to N_2. Here, we need to prove that, if φ is an α-morphism, then $\varphi^{\mathcal{C}}$ is also an α-morphism, as needed in Section 5.

Proposition 1. *Let* $\varphi : N_1 \rightarrow N_2$ *be an* α*-morphism, then* $\varphi^{\mathcal{C}}$ *is an* α*-morphism from* $N_1^{\mathcal{C}}$ *to* N_2*.*

Given an ω-morphism from N_1 to N_2, take $N_1^{\mathcal{C}}$, N_2 and $\varphi^{\mathcal{C}}$. Now, restrict $\varphi^{\mathcal{C}}$ to all the nodes of $N_1^{\mathcal{C}}$ that are not in a bubble $N_1^{-rep}(b_2)$ for some $b_2 \in B_2$ and call it $(\varphi^{\mathcal{C}})^{rep}$: this is a $\widehat{N}$-morphism.

Proposition 2. $((\varphi^{\mathcal{C}})^{rep} \downarrow B_1^{\mathcal{C}}, (\varphi^{\mathcal{C}})^{rep} \downarrow E_1^{\mathcal{C}})$ *is a* $\widehat{N}$*-morphism.*

Every α-morphism is obviously an ω-morphism. Adding the representation for each condition of N_2 does not modify its behaviour, because of the constraint on sequential components. Hence, the results achieved here hold for α-morphisms. In this sense, we consider them as a special case of $\widehat{N}$-morphisms.

The converse is not true, as shown in Fig. 3, where an $\widehat{N}$-morphism from N_1 to N_2 is given by identical names of elements; it is easy to see that there is no α-morphism from N_1 to N_2, since there is no way to map b_3 and b_5.

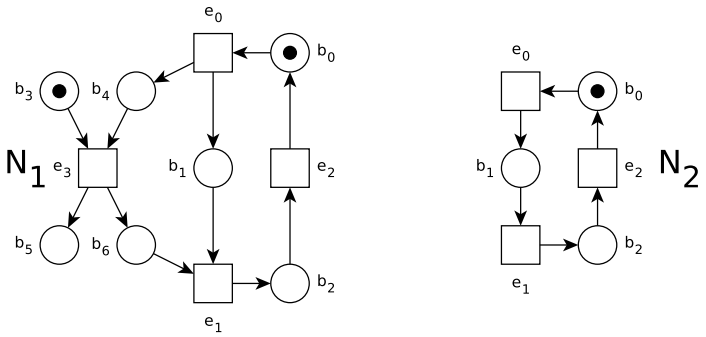


Fig. 3: An example of $\widehat{N}$-morphism which is not an α-morphism

$\widehat{N}$-morphisms are suitable to drive an operation of *composition* of nets. Let N_1 and N_2 be a pair of EN Systems, each one related to another EN System, called interface N_I, by $\widehat{N}$-morphisms, (β_i, η_i). We can see N_I as the protocol of the interaction between them. The morphisms are surjective so that each system cannot ignore a part of the protocol. The composition of N_1 and N_2 on the interface N_I, denoted $N = N_1\langle N_I\rangle N_2$, is given by the union of the local part of each system N_i and the common part corresponding to the protocol. The composition induces $\widehat{N}$-morphisms, (β_i', η_i'), from the composed system to its components.

This composition has several properties, proved in [5], which will be used later and which we informally resume here:

n1 *if the components reflect the sequences of the interface*, the composed net reflects the sequences of the two components;

n2 *if one component is weakly bisimilar to the interface*, then the composed net is weakly bisimilar to the other component.

In particular, **n2** says that if a component is bisimilar to the interface, then only the other component can add behavioural constraints to the composed system.

5 Composition based on α-morphisms

In this section, we define a way of composing SMD-EN systems, in a similar way as in [5], but based on α-morphisms.

The starting point is a set of three SMD-EN systems; one of them, N_I, plays the role of an interface between the other two, N_1 and N_2. A pair of α-morphisms, one from N_1 to N_I, the other from N_2 to N_I, determine how the two components refine the local states of the interface, and which events in the two components have to synchronize.

The crucial point in the definition concerns the choice of synchronizing events. Suppose that the morphisms onto the interface map bubbles A_1 and A_2 to the

same local state b (where A_i is taken in N_i). Then, the representations of A_1 and A_2 are local states which are identified in composing the two nets. This implies that any event in N_1 which puts a token in the representation of A_1 must be synchronized with any event doing the same in the representation of A_2. This explains the definition of the sets E_{sync}, below.

It is assumed that N_1, N_2 and N_I are disjoint and that N_1 and N_2 are canonical with respect to the corresponding morphisms.

Definition 8. *Let $N_i = (B_i, E_i, F_i, m_0^i)$ be an SMD-EN System for $i = 1, 2, I$. Let φ_i, with $i = 1, 2$, be an α-morphism from N_i to N_I. Let N_i be canonical with respect to φ_i.*

We define $N = N_1\langle N_I\rangle N_2 = (B, E, F, m_0)$ such that

$$B = \bigcup_{b_I \in B_I} B_{Bubble(b_I)} \qquad E = \left(\bigcup_{e_I \in E_I} E_{sync}(e_I)\right) \cup \left(\bigcup_{b_I \in B_I} E_{Bubble(b_I)}\right)$$

$$F = \bigcup_{b_I \in B_I} \left(F(b_I) \cup F_{Bubble(b_I)}\right)$$

Where:

$$E_{sync}(e_I) = \{e = \langle e_1, e_2\rangle : e_1 \in E_1, e_2 \in E_2, \varphi_1(e_1) = e_I = \varphi_2(e_2)\}$$

Let $b_I \in B_I$:

$$\begin{aligned} Bubble(b_I) = ((&B_{N_1^{-rep}(b_I)} \cup \{b_I\} \cup B_{N_2^{-rep}(b_I)}),\\ &(E_{N_1^{-rep}(b_I)} \cup E_{N_2^{-rep}(b_I)}),\\ &(F_{N_1^{-rep}(b_I)} \cup F_{N_2^{-rep}(b_I)})) \end{aligned}$$

$$F(b_I) = {}^\bullet F(b_I) \cup F^\bullet(b_I)$$

Let $e = \langle e_1, e_2\rangle \in \bigcup_{e_I \in {}^\bullet b_I} E_{sync}(e_I)$,

$$\begin{aligned} {}^\bullet F(b_I) = &\{(e, b) : b \in {}^\bigcirc Bubble(b_I), (e_1, b) \in F_1\} \cup\\ &\{(e, b_I)\} \cup\\ &\{(e, b) : b \in {}^\bigcirc Bubble(b_I), (e_2, b) \in F_2\} \end{aligned}$$

Let $e = \langle e_1, e_2\rangle \in \bigcup_{e_I \in b_I{}^\bullet} E_{sync}(e_I)$,

$$\begin{aligned} F^\bullet(b_I) = &\{(b, e) : b \in Bubble(b_I)^\bigcirc, (b, e_1) \in F_1\} \cup\\ &\{(b_I, e)\} \cup\\ &\{(b, e) : b \in Bubble(b_I)^\bigcirc, (b, e_2) \in F_2\} \end{aligned}$$

Note that in order to simplify the notation, $N_1\langle N_I\rangle N_2$ does not refer to the morphisms φ_i. By construction, $N = N_1\langle N_I\rangle N_2$ as defined above is an EN System. Moreover, it is covered by sequential components. To see this, take $b \in B$. If $b \in B_I$, then b belongs to a sequential component in N_I, and all the conditions in this component are also in N, and these, together with their neighbourhood, identify a sequential component in N. If $b \in B_i$, then b belongs to a sequential component in N_i, and all the conditions in this component have a corresponding condition in N. It is easy to check that these, together with their neighbourhood, identify a sequential component in N.

We now define a map φ_i' from N onto N_i, and we will show in Theorem 1 that it is an α-morphism.

Definition 9. *Define φ_i' as follows, for each $x \in X$:*

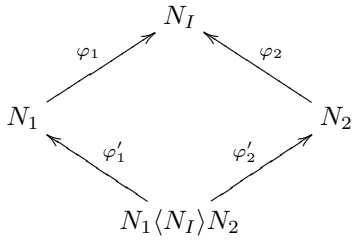
$$\varphi_i'(x) = \begin{cases} x, & \text{if } x \in X_i \\ r_{N_i}(x), & \text{if } x \in B_I \\ r_{N_i}(\varphi_{3-i}(x)), & \text{if } x \in B_{3-i} \\ e_i, & \text{if } x = \langle e_1, e_2\rangle \\ r_{N_i}(\varphi_{3-i}(x)), & \text{if } x \in E_{3-i} \end{cases}$$

Theorem 1. *The map φ_i' is an α-morphism from $N = N_1\langle N_I\rangle N_2$ to $N_i, i = 1, 2$.*

By construction we get the following result:

Proposition 3. *The system $N = N_1\langle N_I\rangle N_2$ is canonical with respect to φ_1' and to φ_2'.*

These results say that the composed system refines both the components, as well as the interface. For each abstract condition there is a corresponding condition in the composed system.



To show that the diagram above commutes, we prove that the operation essentially coincides with the composition based on $\widehat{N}$-morphisms. Since in that case the diagram commutes, the same holds for α-morphisms.

The following proposition is the direct consequence of the definitions of composition.

Proposition 4. *Let $N_i = (B_i, E_i, F_i, m_0^i)$ be an SMD-EN System for $i = 1, 2, I$. Let φ_i, with $i = 1, 2$, be an α-morphism from N_i to N_I. Let N_i be canonical with respect to φ_i. Let $N^\alpha = N_1\langle N_I\rangle^\alpha N_2 = (B, E, F, m_0)$ be the composition of N_1 and N_2 using φ_1 and φ_2. Let φ_i' be the α-morphism from N to N_i created by the composition operation.*

Now, consider the $\widehat{N}$-morphism $((\varphi_i)^{rep} \downarrow B_i, (\varphi_i)^{rep} \downarrow E_i)$. Let $N^{\widehat{N}} = N_1\langle N_I\rangle^{\widehat{N}} N_2 = (B, E, F, m_0)$ be the composition of N_1 and N_2 using $((\varphi_1)^{rep} \downarrow B_1, (\varphi_1)^{rep} \downarrow E_1)$ and $((\varphi_2)^{rep} \downarrow B_2, (\varphi_2)^{rep} \downarrow E_2)$. Let (β_i', η_i') be the $\widehat{N}$-morphism from N to N_i created by the composition operation.

The systems N^α and $N^{\widehat{N}}$ are isomorphic, $\beta_i' = (\varphi_i')^{rep} \downarrow B_i$ and $\eta_i' = (\varphi_i)^{rep} \downarrow E_i$.

The diagram in Fig. 1 is an example of composition which is not a pull-back diagram. It is still an open problem whether, in general, the diagram of a composition operation is a pushout.

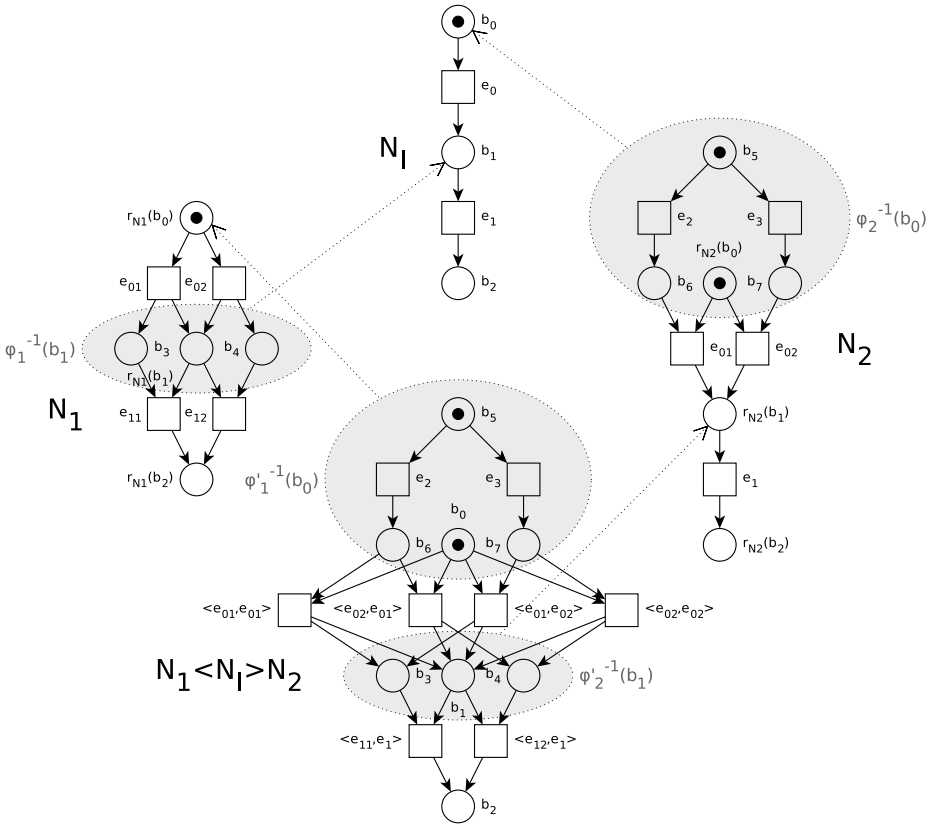


Fig. 4: An example of composition based on α-morphisms

From results in Section 3 and 4 we can derive a property valid for composition based on α-morphisms. We know that, if N_1 is weakly bisimilar to N_I then N is weakly bisimilar to N_2. By **p7** we can check weak bisimilarity between N_1 and N_I using **c1**, **c2** and **c3**. These constraints are either structural or locally behavioural, while, in the case of $\widehat{N}$-morphisms, checking bisimilarity must be made globally. Fig. 4 shows an example in which N_1 and N_2 are weakly bisimilar to N_I. Hence $N_1\langle N_I\rangle N_2$ is weakly bisimilar to N_1, N_2 and N_I.

6 Conclusions

We have proposed a way to compose State Machine Decomposable EN Systems, by identifying elements of the components. The identification is ruled by morphisms from the components to a net, which can be seen as an interface or as a common abstraction of the overall system.

We have proved that α-morphisms can be seen as a particular case of $\widehat{N}$-morphisms [5] and that, composing two systems using α-morphisms or using $\widehat{N}$-morphisms, we obtain isomorphic systems.

Here, we have looked at the properties of the composed net which can be deduced from properties of the components. In particular, the constraints of α-morphisms allow to check bisimilarity between a component and the interface by using only structural and local behavioural constraints. By a property holding also in the case of $\widehat{N}$-morphisms, this can be lifted to bisimilarity between the composed net and the components.

We plan to explore the extension of these ideas to P/T nets and to colored nets that can be unfolded to State Machine Decomposable EN Systems.

Acknowledgments

Work partially supported by MIUR.

References

1. Marek A. Bednarczyk, Luca Bernardinello, Benoît Caillaud, Wiesław Pawłowski, and Lucia Pomello. Modular system development with pullbacks. In Wil M. P. van der Aalst and Eike Best, editors, *ICATPN*, volume 2679 of *Lecture Notes in Computer Science*, pages 140–160. Springer, 2003.
2. Marek A. Bednarczyk and Andrzej M. Borzyszkowski. On concurrent realization of reactive systems and their morphisms. In Hartmut Ehrig, Gabriel Juhás, Julia Padberg, and Grzegorz Rozenberg, editors, *Unifying Petri Nets*, volume 2128 of *Lecture Notes in Computer Science*, pages 346–379. Springer, 2001.
3. Luca Bernardinello, Elisabetta Mangioni, and Lucia Pomello. Composition of elementary net systems based on α-morphisms. Internal report (2012), available at `http://www.mc3.disco.unimib.it/pub/bmp2012-compo.pdf`.
4. Luca Bernardinello, Elisabetta Mangioni, and Lucia Pomello. Local state refinement on elementary net systems: an approach based on morphisms. In Proc. Workshop PNSE 2012, Hamburg 2012.

5. Luca Bernardinello, Elena Monticelli, and Lucia Pomello. On preserving structural and behavioural properties by composing net systems on interfaces. *Fundam. Inform.*, 80(1-3):31–47, 2007.
6. Wilfried Brauer, Robert Gold, and Walter Vogler. A survey of behaviour and equivalence preserving refinements of Petri nets. *Advances in Petri Nets 1990*, pages 1–46, 1991.
7. Jörg Desel and Agathe Merceron. Vicinity respecting homomorphisms for abstracting system requirements. *Transactions on Petri Nets and Other Models of Concurrency*, 4:1–20, 2010.
8. Eric Fabre. On the construction of pullbacks for safe Petri nets. In Susanna Donatelli and P. S. Thiagarajan, editors, *ICATPN*, volume 4024 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2006.
9. Claude Girault and Rüdiger Valk. *Petri nets for systems engineering - a guide to modeling, verification, and applications*. Springer, 2003.
10. Mogens Nielsen, Grzegorz Rozenberg, and P. S. Thiagarajan. Elementary transition systems. *Theor. Comput. Sci.*, 96(1):3–33, 1992.
11. Mogens Nielsen, Grzegorz Rozenberg, and P. S. Thiagarajan. Elementary transition systems and refinement. *Acta Inf.*, 29(6/7):555–578, 1992.
12. Julia Padberg and Milan Urbásek. Rule-based refinement of Petri nets: A survey. In Hartmut Ehrig, Wolfgang Reisig, Grzegorz Rozenberg, and Herbert Weber, editors, *Petri Net Technology for Communication-Based Systems*, volume 2472 of *Lecture Notes in Computer Science*, pages 161–196. Springer, 2003.
13. Lucia Pomello and Luca Bernardinello. Formal tools for modular system development. In Jordi Cortadella and Wolfgang Reisig, editors, *ICATPN*, volume 3099 of *Lecture Notes in Computer Science*, pages 77–96. Springer, 2004.
14. Lucia Pomello, Grzegorz Rozenberg, and Carla Simone. A survey of equivalence notions for net based systems. In Grzegorz Rozenberg, editor, *Advances in Petri Nets: The DEMON Project*, volume 609 of *Lecture Notes in Computer Science*, pages 410–472. Springer, 1992.
15. Grzegorz Rozenberg and Joost Engelfriet. Elementary net systems. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 12–121. Springer, 1996.
16. Walter Vogler. Executions: A new partial-order semantics of Petri nets. *Theor. Comput. Sci.*, 91(2):205–238, 1991.
17. Glynn Winskel. Petri nets, algebras, morphisms, and compositionality. *Inf. Comput.*, 72(3):197–238, 1987.

Deciding the Precongruence for Deadlock Freedom Using Operating Guidelines

Richard Müller[1,2] and Christian Stahl[2]

[1] Institut für Informatik, Humboldt-Universität zu Berlin, Germany
richard.mueller@informatik.hu-berlin.de
[2] Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, The Netherlands
c.stahl@tue.nl

Abstract. In the context of asynchronously communicating and deadlock free services, the refinement relation of services has been formalized by the *accordance preorder*. A service *Impl* accords with a service *Spec* if every *controller* of *Spec*—that is, every environment that can interact with service *Spec* without deadlocking—is a controller of *Impl*. The procedure to decide accordance of two services uses that the set of controllers of a finite-state service has a finite representation, called *operating guideline*. Recently, it has been shown that the accordance preorder is not a precongruence and thus the decision procedure based on operating guidelines cannot be used. In this paper, we *adapt the results on operating guidelines to the precongruence setting*: We define an operating guideline that represents all controllers of a service w.r.t. the accordance precongruence and show how this refinement relation of two services can be decided based on their operating guidelines.

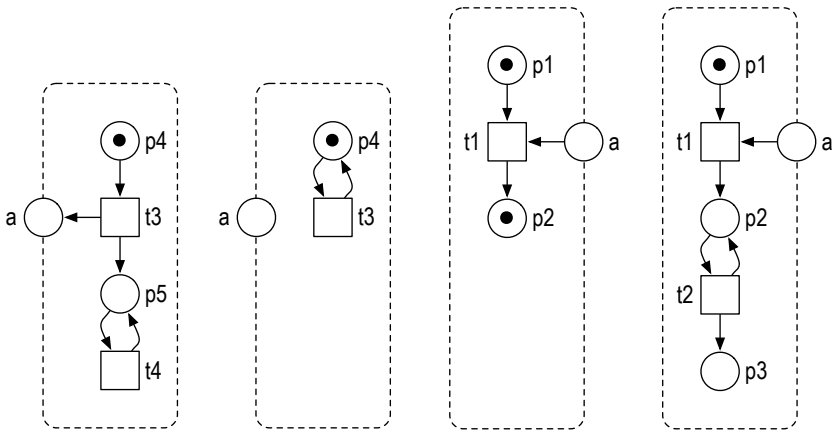
1 Introduction

Service-oriented computing (SOC) [6] aims at building complex systems by aggregating less complex, independently-developed building blocks called *services*. A service is an autonomous system that has an interface to interact with other services via asynchronous message passing. Designing a system in such a way allows for rapidly adjusting it to prevalent needs. Services sometimes need to be replaced—for example, when new features have been implemented or bugs have been fixed. This requires a notion of service *refinement*, which should, according to the idea of SOC, respect *compositionality*: If a service *Impl* refines a service *Spec*, then any environment that can correctly interact with *Spec* can also correctly interact with *Impl*. We refer to such an environment as a *controller* of *Impl* and *Spec*, respectively. Compositionality is crucial, because organizations usually do not know the services of other organizations involved in the system.

The absence of deadlocks is a commonly agreed minimal requirement for the behavioral correctness of a service-oriented system. Stahl et al. [7] formalized the replacement (or refinement) relation in the context of deadlock freedom by the *accordance* preorder. The decision procedure uses that, for finite-state

services with bounded buffers, the set of controllers has a finite representation, the *operating guideline* [4] of the service. The decision procedure in [7] has two inherent characteristics: First, the interior of a service must be bounded when considered in isolation. Second, it allows for two possibly different bounds: one for the buffers and one for the interior of a service.

Recently, Stahl and Vogler [8] introduced a modified accordance relation which differs from the original accordance relation in two ways: First, the modified accordance relation has been proven to be a *precongruence* w.r.t. service composition; that is, it respects compositionality. Second, the modified accordance relation is more uniform than the original accordance relation in [7]: Stahl and Vogler [8] do not require the interior of a service to be bounded when considered in isolation and prescribe only one bound for the buffers and for the interior of a service rather than possible different bounds as in [7].



(a) Open net *Spec* **(b)** Open net *Impl* **(c)** Open net N_1 **(d)** Open net N_2

Fig. 1. Open net *Impl* accords with open net *Spec* but not vice versa.

We illustrate the difference between the accordance relation in [7] and the precongruence in [8] with an example: Figure 1 depicts four services modeled as open nets. As shown in [8], open net *Impl* accords with open net *Spec* for a bound $b = 1$ if we consider the precongruence, but *Spec* does not accord with *Impl*. To see this, consider the open net N_1 in Fig. 1(c) and compose N_1 with *Spec* and *Impl* by merging the common interface places a. The composition of *Impl* and N_1 has only one reachable marking, $[p_1, p_2, p_4]$, in which transition t_3 is continuously enabled. Thus, the composition is deadlock free and N_1 is a controller of *Impl*. Now consider the composition of *Spec* and N_1. It has a reachable marking where p_2 contains two tokens. Thus, the composition is not 1-bounded and N_1 is not a controller (for a bound of 1) of *Spec*. Similarly, open

net N_2 in Fig. 1(d) is a controller of *Impl* but not a controller of *Spec* (for a bound of 1), because p_3 is unbounded in the composition of *Spec* and N_2.

However, applying the decision procedure in [7] based on operating guidelines, *Spec* and *Impl* are even accordance equivalent (assuming a single bound for the interface and the interior); that is, every controller of *Impl*—like the open net N_1 or N_2—is also a controller of *Spec*. The cause for this result is that [7] does not consider N_1 and N_2, because their interiors are not 1-bounded.

So the example shows, if we assume a single bound for the interface and the interior of a service, then the accordance precongruence implies accordance but not the other way around. The reason is that the precongruence is more uniform and considers a more general notion of a service. If we consider different bounds for the interface and the interior of a service, then both refinement relations are incomparable.

Stahl and Vogler [8] presented a procedure to decide the accordance precongruence, but they also showed that the accordance precongruence cannot be decided using the procedure in [7] based on operating guidelines without adaptation. In this paper, we present an operating guideline representing the set of all controllers in the precongruence setting of [8] and show how this operating guideline can be used to decide accordance of two services. Our motivation for adapting the theory of operating guidelines from the setting of [7] to the setting of [8] is twofold: First, we want to present the theory for deciding accordance using operating guidelines such that the existing implementation in the tool Cosme [5] can be reused and that the technique can also be applied in the precongruence setting. Second, operating guidelines have proved their usefulness also in other applications than deciding accordance, including service correction [3], test case generation [1], and instance migration [2]. As the more general notion of a controller is advantageous also for those applications, extending the theory on operating guidelines is natural.

This paper is organized as follows: Section 2 introduces open nets, our formal model for services, and gives some background information. Section 3 introduces operating guidelines and adapts the matching technique to the modified accordance relation. Section 4 decides the precongruence for deadlock freedom using operating guidelines. We close with a discussion of related work and a conclusion in Sect. 5.

2 Preliminaries

This section provides the basic notions, such as Petri nets, open nets for modeling services, and open net environments for describing the behavior of open nets.

For two sets A and B, let $A \uplus B$ denote the disjoint union; writing $A \uplus B$ expresses the implicit assumption that A and B are disjoint. Let $\mathbb{N}$ denote the non-negative integers, and let $\mathbb{N}^+$ denote the positive integers. For a set A, let $\mathcal{P}(A)$ denote the powerset of A, and let $|A|$ denote the cardinality of A.

2.1 Petri Nets

As a basic model, we use place/transition Petri nets extended with a set of final markings and transition labels.

Definition 1 (net). A *net* $N = (P, T, F, m_N, \Omega)$ consists of

- a finite set P of *places*,
- a finite set T of *transitions* such that P and T are disjoint,
- a *flow relation* $F \subseteq (P \times T) \uplus (T \times P)$,
- an *initial marking* m_N, where a marking is a mapping $m : P \to \mathbb{N}$, and
- a set Ω of *final markings*.

A *labeled net* $N = (P, T, F, m_N, \Omega, \Sigma_{in}, \Sigma_{out}, l)$ is a net (P, T, F, m_N, Ω) together with an *alphabet* $\Sigma = \Sigma_{in} \uplus \Sigma_{out}$ of *input actions* Σ_{in} and *output actions* Σ_{out} and a *labeling function* $l : T \to \Sigma \uplus \{\tau\}$, where τ represents an invisible, internal action.

In this paper, we only treat labeled nets where, for every transition t, the *label* $l(t)$ of t is either τ or t itself.

Introducing net N implicitly introduces its components P, T, F, m_N, Ω; the same applies to nets N', N_1, etc. and their components $P', T', F', m_{N'}, \Omega'$, and $P_1, T_1, F_1, m_{N_1}, \Omega_1$, respectively—and it also applies to other structures later on.

Graphically, a circle represents a place, a box represents a transition, and the directed arcs between places and transitions represent the flow relation. A marking is a distribution of tokens over the places. Graphically, a black dot represents a token. Transition labels beside τ are written into the respective boxes.

Let $x \in P \uplus T$ be a node of a net N. As usual, ${}^\bullet x = \{y \mid (y, x) \in F\}$ denotes the *preset* of x and $x^\bullet = \{y \mid (x, y) \in F\}$ the *postset* of x. We canonically extend the notion of a preset/postset to sets of nodes. We interpret presets and postsets as multisets when used in operations also involving multisets. A marking is a multiset over the set P of places; for example, $[p_1, 2p_2]$ denotes a marking m with $m(p_1) = 1$, $m(p_2) = 2$, and $m(p) = 0$ for $p \in P \setminus \{p_1, p_2\}$. For $n \in \mathbb{N}$, a place $p \in P$ and a set M of markings over P, $M(p) = n$ denotes that for all $m \in M$, $m(p) = n$. We define $+$ and $-$ for the sum and the difference of two markings and $=, <, >, \leq, \geq$ for comparison of markings in the standard way. We canonically extend the notion of a marking of N to supersets $Q \supseteq P$ of places; that is, for a mapping $m : P \to \mathbb{N}$, we extend m to the marking $m : Q \to \mathbb{N}$ such that for all $p \in Q \setminus P$, $m(p) = 0$. Analogously, a marking can be restricted to a subset $Q \subseteq P$ of the places of N.

The *behavior* of a net N relies on the marking of N and changing the marking by the firing of transitions of N. A transition $t \in T$ is *enabled* at a marking m, denoted by $m \xrightarrow{t}$, if for all $p \in {}^\bullet t$, $m(p) > 0$. If t is enabled at m, it can *fire*, thereby changing the marking m to a marking $m' = m - {}^\bullet t + t^\bullet$. The firing of t is denoted by $m \xrightarrow{t} m'$; that is, t is enabled at m and firing it results in m'. The behavior of N can be extended to sequences: $m_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{k-1}} m_k$ is a *run* of

N if for all $0 < i < k$, $m_i \xrightarrow{t_i} m_{i+1}$. A marking m' is *reachable from* a marking m if there exists a (possibly empty) run $m_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{k-1}} m_k$ with $m = m_1$ and $m' = m_k$; for $v = t_1 \ldots t_k$, we also write $m_1 \xrightarrow{v} m_k$. Marking m' is *reachable* if $m_N = m$. The set M_N represents the set of all reachable markings of N.

In the case of labeled nets, we lift runs to traces: If $m_1 \xrightarrow{v} m_k$ and w is obtained from v by replacing each transition by its label and removing all τ labels, we write $m_1 \stackrel{w}{\Longrightarrow} m_k$ and refer to w as a *trace*. As usual, ε denotes the empty trace. The *reachability graph* $RG(N)$ of net N has the reachable markings M_N as its nodes and a t-labeled edge from m to m' whenever $m \xrightarrow{t} m'$ in N. In the case of a labeled net, each edge label t is replaced by $l(t)$.

Finally, we introduce b-boundedness and deadlock freedom of nets. A marking m of net N is *b-bounded* for a bound $b \in \mathbb{N}^+$, if $m(p) \leq b$ for all $p \in P$. Net N is *b-bounded* if every reachable marking is b-bounded. The set M_N^b represents the set of all reachable b-bounded markings of N. A reachable marking $m \notin \Omega$ of N is a *deadlock* if no transition $t \in T$ of N is enabled at m. If N has no deadlock, then it is deadlock free.

2.2 Open Nets and Open Net Behavior

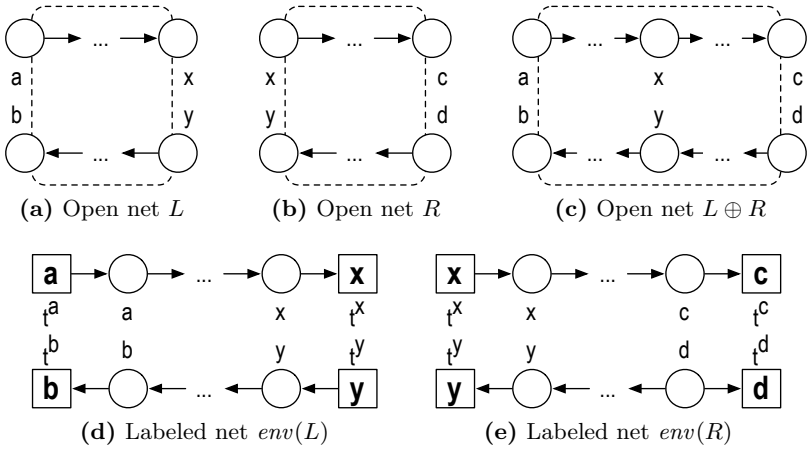
Like Lohmann et al. [4] and Stahl et al. [7], we model services as *open nets* [9,4], thereby restricting ourselves to the communication protocol of a service. In the model, we abstract from data and identify each message by the label of its message channel. An open net extends a net by an interface. An interface consists of two disjoint sets of input and output places corresponding to asynchronous input and output channels. In the initial marking and the final markings, interface places are not marked. An input place has an empty preset, and an output place has an empty postset.

Definition 2 (open net). An *open net* N is a tuple $(P, T, F, m_N, \Omega, I, O)$ with

- $(P \uplus I \uplus O, T, F, m_N, \Omega)$ is a net,
- for all $p \in I \uplus O$, $m_N(p) = 0$ and $\Omega(p) = 0$,
- the set I of *input places* satisfies ${}^\bullet I = \emptyset$, and
- the set O of *output places* satisfies $O^\bullet = \emptyset$.

If $I = O = \emptyset$, then N is a *closed net*. Open net N is *sequentially communicating* if each transition is connected to at most one *interface place* $I \uplus O$. The *inner* net *inner*(N) results from removing the interface places and their adjacent arcs from N. Two open nets are *interface equivalent* if they have the same sets of input and output places.

Graphically, we represent an open net like a net with a dashed frame around it. The interface places are depicted on the frame. Later, we consider the behavior of an open net, which is basically its reachability graph. To simplify the labeling of transitions connected to interface places, we only consider sequentially communicating nets. That way, each transition is labeled by a single label



(a) Open net L **(b)** Open net R **(c)** Open net $L \oplus R$

(d) Labeled net $env(L)$ **(e)** Labeled net $env(R)$

Fig. 2. Schematic example of open nets, open net composition, and their environment.

rather by a set of labels. This restriction is not significant as every open net can be transformed into an equivalent sequentially communicating open net [4].

For the composition of open nets, we assume that the sets of transitions are pairwise disjoint and that no internal place of an open net is a place of any other open net. In contrast, the interfaces intentionally overlap. We require that all communication is *bilateral* and *directed*; that is, every shared place p has only one open net that sends into p and one open net that receives from p. We refer to open nets that fulfill these properties as *composable*. We compose two composable open nets N_1 and N_2 by merging shared interface places and turn these places into internal places; see Fig. 2(a) and 2(b) for a schematic example of open nets and their composition. The definition of composable thereby guarantees that an open net composition is again an open net (possibly a closed net).

Definition 3 (open net composition). Open nets N_1 and N_2 are *composable* if $(P_1 \uplus T_1 \uplus I_1 \uplus O_1) \cap (P_2 \uplus T_2 \uplus I_2 \uplus O_2) = (I_1 \cap O_2) \uplus (I_2 \cap O_1)$. The *composition* of two composable open nets N_1 and N_2 is the open net $N_1 \oplus N_2 = (P, T, F, m_N, \Omega, I, O)$ where

- $P = P_1 \uplus P_2 \uplus (I_1 \cap O_2) \uplus (I_2 \cap O_1)$,
- $T = T_1 \uplus T_2$,
- $F = F_1 \uplus F_2$,
- $m_N = m_{N_1} + m_{N_2}$,
- $I = (I_1 \uplus I_2) \setminus (O_1 \uplus O_2)$,
- $O = (O_1 \uplus O_2) \setminus (I_1 \uplus I_2)$, and
- $\Omega = \{m_1 + m_2 \mid m_1 \in \Omega_1, m_2 \in \Omega_2\}$.

To define the *behavior* of an open net N, we consider its environment $env(N)$. The net $env(N)$ is a net that can be constructed from N by adding to each

interface place $p \in I \uplus O$ a p-labeled transition t^p in $env(N)$. The net $env(N)$ is just a tool to define our characterizations and prove our results. Intuitively, one can understand the construction as translating the asynchronous interface of N into a buffered synchronous interface (with unbounded buffers) described by the transition labels of $env(N)$.

Definition 4 (open net environment). The *environment* of an open net N is the labeled net $env(N) = (P \uplus I \uplus O, T \uplus T', F \uplus F', m_N, \Omega, I, O, l)$ where

- $T' = \{t^x \mid x \in I \uplus O\}$ is the set of *interface transitions*,
- $F' = \{(t^x, x) \mid x \in I\} \uplus \{(x, t^x) \mid x \in O\}$, and
- $l(t) = \begin{cases} \tau, & t \in T \\ x, & t^x \in T'. \end{cases}$

We refer to a transition from T as *internal transition*. A marking m of $env(N)$ is *stable* if at most internal transitions of $env(N)$ are enabled at m.

Figures 2(d) and 2(e) show the environments of the open nets L and R from Fig. 2(a) and 2(b). A transition label is depicted inside a transition with bold font to distinguish it from the transition's identity.

The behavior of an open net N can now be defined by the reachability graph $RG(env(N))$ of its environment. As we are interested in finite-state services, we always define the behavior of an open net with regard to a bound b. As soon as b is violated, we can stop the computation of the behavior in this state; however, we keep this state to identify the bound violation.

Definition 5 (open net behavior). Let $b \in \mathbb{N}^+$. The *b-behavior* $beh_b(N)$ of an open net N is the reachability graph of $env(N)$ where we remove all outgoing edges from every non-b-bounded node (thereby removing unreachable nodes and edges too).

Clearly, the b-behavior of an open net N has at most $(b+2)^{(|P|+|I|+|O|)}$ states.

Figure 3 depicts the environment net of open net N_2 and its behavior $beh_1(N_2)$. Recall that transitions t_1 and t_2 are labeled τ. Every leaf in $beh_1(N_2)$ violates the bound and has thus no successor.

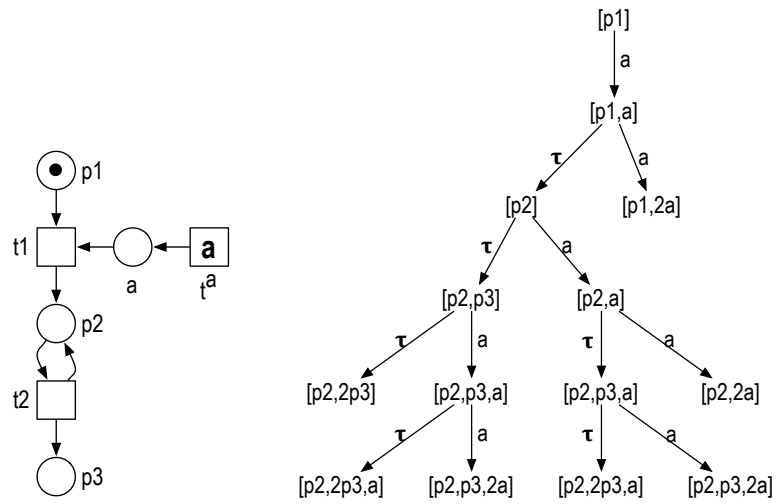
We interpret $beh_b(N)$ as a labeled automaton with input and output labels.

Definition 6 (automaton). An *automaton* $A = (Q, E, q_A, \Sigma_{in}, \Sigma_{out})$ consists of

- a finite set Q of *states*,
- an *edge relation* $E \subseteq Q \times \big(\Sigma_{in} \uplus \Sigma_{out} \uplus \{\tau\}\big) \times Q$,
- an *initial node* q_A, and
- an *alphabet* $\Sigma = \Sigma_{in} \uplus \Sigma_{out}$ of *input labels* Σ_{in} and *output labels* Σ_{out}.

A is *deterministic* if no node has two outgoing edges with the same label.

We compare two automata with a simulation relation, thereby treating τ as an ordinary action.



(a) Labeled net $env(N_2)$ **(b)** Behavior $beh_1(N_2)$

Fig. 3. Constructing the 1-behavior of open net N_2.

Definition 7 (simulation relation). Let A and B be two automata with label set $\Sigma = \Sigma_{in} \uplus \Sigma_{out}$. Then $\varrho \subseteq Q_A \times Q_B$ is a *simulation* of A by B if

- $(q_A, q_B) \in \varrho$, and
- for every $(p, q) \in \varrho$, $x \in \Sigma \uplus \{\tau\}$, $p' \in Q_A$ such that $p \xrightarrow{x} p'$ in A, there exists $q' \in Q_B$ such that $q \xrightarrow{x} q'$ in B and $(p', q') \in \varrho$.

Simulation ϱ is *minimal* if for every simulation ϱ' of A by B, $\varrho \subseteq \varrho'$.

For all automata A and B where B is deterministic, the minimal simulation relation of A by B is uniquely defined.

3 Operating Guidelines

In this section, we formally define the notion of a controller of an open net N and present a finite representation of all controllers of N, the *operating guideline* of N.

The composition of a service C with a service N shall be deadlock free; that is, if the composition gets stuck, then it is in a final state. As we are interested in finite-state services, the composition must be bounded. A service C guaranteeing these two requirements can be seen as a *controller* of the service N.

Definition 8 (b-controller). Let $b \in \mathbb{N}^+$. An open net C is a *b-controller* of an open net N if the composition $N \oplus C$ is a closed net, deadlock free, and b-bounded.

A b-operating guideline $OG_b(N)$ of a service N describes how another service C should successfully communicate with N. Technically, it characterizes the possibly infinite set of b-controllers of N in a finite manner. Because a b-controller of N provides suitable inputs for N and accepts its outputs, $OG_b(N)$ interchanges the inputs and outputs of N. The structure of $OG_b(N)$ is an automaton where a Boolean formula is attached to each state. The structure is the behavior of a b-controller that exhibits the behavior of every b-controller of N; the formula of a state indicates which combinations of outgoing edges must be present in any b-controller. Thus, a literal of such a Boolean formula is a transition label of N or the literal *final*, specifying that N is in a final state. That way, we can employ simulation for comparing the behavior of an open net with $OG_b(N)$ later on.

Definition 9 (annotated automaton). An *annotated automaton* $(Q, E, q_A, \Sigma_{in}, \Sigma_{out}, \phi)$ is an automaton $(Q, E, q_A, \Sigma_{in}, \Sigma_{out})$ whose nodes $q \in Q$ are annotated with a *Boolean formula* $\phi(q)$ over $\Sigma_{in} \uplus \Sigma_{out} \uplus \{final\}$.

To construct $OG_b(N)$, we calculate the b-behavior $beh_b(N)$ of N and make the automaton deterministic by constructing the powerset automaton. A state of $OG_b(N)$ contains a set of markings of $env(N)$; we refer to it as a *node*. These markings can be reached by firing internal transitions of $env(N)$. An edge connects two nodes of $OG_b(N)$, thereby referring to an interface transition of $env(N)$ (i.e., the environment takes a token from an output place or produces a token on an input place of N). A b-controller cannot know which marking m of a node Q net $env(N)$ might be in, but it has to avoid a deadlock and a bound violation in any case; the formula $\phi(Q)$ describes how to do this. The literals of ϕ are $I \uplus O \uplus \{final\}$. Recall that nonstable markings have an internal transition enabled and, thus, are not deadlocks; all internal transitions remain in the same node. As a consequence, $\phi(Q)$ is a *conjunction indexed by all stable markings* $m \in Q$. Every conjunct is a disjunction of the following propositional atoms: *final* if m is a final marking, $x \in I$ if $Q \xrightarrow{x}$ (i.e., x does not lead to a bound violation in any case), and $x \in O$ if t^x is enabled at m (i.e., if in marking m, net N has already produced a message on output place x). Hence, the formulae are in conjunctive normal form (CNF) without negation. Here, $Q \xrightarrow{x}$ means that Q has an outgoing x-labeled edge.

Definition 10 (b-operating guideline). Let $b \in \mathbb{N}^+$. The *b-operating guideline* of an open net N is the annotated automaton $OG_b(N) = (\mathcal{Q}, E, Q_0, \Sigma_{in}, \Sigma_{out}, \phi)$, where

- $\mathcal{Q} = \mathcal{P}(M^b_{env(N)})$ is a set of *nodes*,
- $E = \{(Q, x, Q') \in \mathcal{Q} \times I \uplus O \times \mathcal{Q} \mid Q' = \{m' \mid \exists m \in Q : m \overset{x}{\Longrightarrow} m'\}\}$ $\uplus \{(Q, \tau, Q) \mid Q \in \mathcal{Q}\}$ is a set of *edges*,
- $Q_0 = \{m' \mid m_{env(N)} \overset{\varepsilon}{\Longrightarrow} m'\} \cap \mathcal{P}(M^b_{env(N)})$ is the *initial node*,
- $\Sigma_{in} = O$ are the *input labels*,
- $\Sigma_{out} = I$ are the *output labels*, and

– ϕ associates to each $Q \in \mathcal{Q}$ a *Boolean formula* with propositional atoms taken from $I \uplus O \uplus \{\mathit{final}\}$ such that

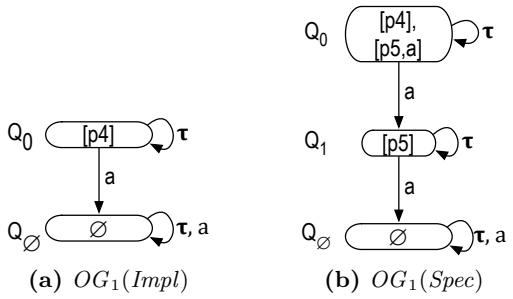
$$\phi(Q) = \bigwedge_{m : m \in Q \wedge m \text{ is stable}} \big(\psi_1(m) \vee \psi_2(m)\big) \quad \text{with}$$

$$\psi_1(m) = \bigvee_{x : x \in I \wedge Q \xrightarrow{x}} x \quad \vee \quad \bigvee_{x : x \in O \wedge m \xrightarrow{t^x}} x$$

$$\psi_2(m) = \begin{cases} \mathit{final}, & \text{if } m \in \Omega_{env(N)}, \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

Clearly, $OG_b(N)$ is finite and deterministic by construction; if $Q_0 = \emptyset$, then the b-operating guideline of N does not exist. We refer to $Q \in \mathcal{Q}$ with $Q = \emptyset$ as the *empty node* and denote it by $Q_\emptyset$. Intuitively, the empty node $Q_\emptyset$ refers to markings which are unreachable in $env(N)$.

We proceed with a short complexity analysis. Let $b \in \mathbb{N}^+$ and N be an open net. Let further $x = |M^b_{env(N)}|$ denote the cardinality of the set of reachable, b-bounded markings of $env(N)$, and let $k = |I \uplus O|$ denote the size of the interface. The powerset construction may yield, in worst case, 2^x nodes of $OG_b(N)$. The formula $\phi(Q)$ of a node Q has at most $x \cdot (k+1)$ literals. As calculating the formula of a node can be done during the construction, $OG_b(N)$ can be computed in time and space proportional to $O(2^x \cdot x \cdot (k+1))$.



(a) $OG_1(\mathit{Impl})$ **(b)** $OG_1(\mathit{Spec})$

Fig. 4. Operating guidelines of open nets *Impl* and *Spec*. The annotation of all nodes is *true*, which we omitted.

Figure 4 depicts the 1-operating guidelines for open nets *Spec* and *Impl*. All nodes of $OG_1(\mathit{Impl})$ and $OG_1(\mathit{Spec})$ have the same annotation, *true*[3], thus we omitted them. For $OG_1(\mathit{Impl})$, we have $Q_0 = \{[p_4]\}$. A 1-controller can receive

[3] An annotation is a formula over $I \uplus O \uplus \{\mathit{final}\}$; *true* and *false* are also Boolean formulae.

message a, but *Impl* will never send this message. Thus, there is an a-labeled edge from Q_0 to the empty node $Q_\emptyset$. In $Q_\emptyset$, every action can occur, because the empty node refers to markings which are unreachable in $env(Impl)$.

We determine if an open net C is a b-controller of an open net N by *matching* its b-behavior $beh_b(C)$ with the b-operating guideline $OG_b(N)$ of N. To this end, we need to check whether C and N are composable, the behavior of C can be mimicked by $OG_b(N)$ (by checking a simulation relation), and every state m of $beh_b(C)$ satisfies the Boolean formula in the corresponding node Q of $OG_b(N)$. State m satisfies $\phi(Q)$ if either a correct combination of interface transition of $env(C)$ is enabled at m such that $N \oplus C$ remains b-bounded or m is a final marking and $env(N)$ is in a final marking, too (i.e., $\phi(Q)$ contains the literal *final*).

Definition 11 (matching). Let $b \in \mathbb{N}^+$ and let N and C be composable open nets. Then $beh_b(C)$ *matches* with $OG_b(N)$ if

1. The input (output) labels of $beh_b(C)$ are the input (output) labels of $OG_b(N)$.
2. There exists a minimal simulation relation ϱ of $beh_b(C)$ by $OG_b(N)$ such that
   (a) if $[m, Q] \in \varrho$ with m not b-bounded in $env(C)$, then $Q = Q_\emptyset$, and
   (b) if $[m, Q] \in \varrho$ with m stable in $env(C)$, then $\phi(Q)$ evaluates to *true*, written $m \models \phi(Q)$, for the following assignment β:
   - $\beta(c) = true$ if $c \neq final$ and $m \xrightarrow{c}$ in $beh_b(C)$,
   - $\beta(c) = true$ if $c = final$ and $m \in \Omega_{env(C)}$, and
   - $\beta(c) = false$, otherwise.

Consider again open net N_2, which is a 1-controller of *Impl*. Automaton $beh_1(N_2)$ in Fig. 3(b)) matches with $OG_1(Impl)$ (Fig. 4(a)). The simulation relation relates state $[p1]$ with Q_0 and all other states of $beh_1(N_2)$ with $Q_\emptyset$. The annotations trivially evaluate to *true*. Open net N_2 is not a 1-controller of *Spec* and $beh_1(N_2)$ does not match with $OG_1(Spec)$: The simulation relation relates state $[p_2, 2p_3]$ with node Q_1, thereby violating item 2(a) of Def. 11.

With the next theorem, we show that the b-operating guideline of an open net N characterizes the set of b-controllers of N.

Theorem 12 (b-controllability vs. matching). *Let $b \in \mathbb{N}^+$. For composable open nets N and C, C is a b-controller of N iff $beh_b(C)$ matches with $OG_b(N)$.*

Proof. ($\Rightarrow$): Let C be a b-controller of N. Then item (1) of Def. 11 holds because C and N are composable and $N \oplus C$ is a closed net.

Suppose a simulation relation ϱ of $beh_b(C)$ by $OG_b(N)$ does not exist. Then there exists $(m, Q) \in \varrho$ and $m \xrightarrow{x}$ in $beh_b(C)$ but $Q \not\xrightarrow{x}$ in $OG_b(N)$ by Def. 7. By Def. 10, $Q \xrightarrow{x} Q'$ and there exists a marking of $env(N)$ in Q' that violates bound b and, therefore, Q' has been removed from $OG_b(N)$. As the respective trace to Q' is also a trace in $beh_b(C)$, there is a corresponding marking in $M_{N \oplus C}$ that violates the bound, and we have a contradiction to our assumption. Thus, ϱ exists, and ϱ is even minimal as $OG_b(N)$ is deterministic by Def. 10.

To show item (2a) of Def. 11, assume $(m, Q) \in \varrho$, with m is not b-bounded, and $Q \neq Q_\emptyset$. There exists $v \in (I \uplus O)^*$ with $m_{env(C)} \stackrel{v}{\Longrightarrow} m$ in $env(C)$ by Def. 5 and $m_{env(N)} \stackrel{v}{\Longrightarrow} m'$ in $env(N)$ by Def. 10. As a consequence, we find a corresponding marking in $M_{N \oplus C}$ that is not b-bounded; thus, we have a contradiction to our assumption and conclude $Q = Q_\emptyset$.

To show item (2b) of Def. 11, let $(m, Q) \in \varrho$ such that m is stable in $env(C)$. We show for each $m' \in Q$ with m' is stable in $env(N)$ that $m \models \psi_1(m') \vee \psi_2(m')$. If $m + m' \in \Omega_{N \oplus C}$, then $m \in \Omega_{env(C)}$ and $\psi_2(m') = \mathit{final}$, thus $m \models \psi_2(m')$ by Def. 11. Assume $m + m' \notin \Omega_{N \oplus C}$. Then C can either produce a token on a place $i \in I_N$ or consume a token from a place $o \in O_N$, because $N \oplus C$ is deadlock free by assumption. In the former case, we have $m \stackrel{i}{\rightarrow}$ in $beh_b(C)$, and $Q \stackrel{i}{\rightarrow}$ as $N \oplus C$ is b-bounded. Thus, $m \models \psi_1(m')$ by Def. 11. In the latter case, we have $m \stackrel{o}{\rightarrow}$ in $beh_b(C)$, and $m' \stackrel{t^o}{\longrightarrow}$. Thus, $m \models \psi_1(m')$ by Def. 11.

($\Leftarrow$): Let ϱ be a minimal simulation of $beh_b(C)$ by $OG_b(N)$. We have to show that $N \oplus C$ is a closed net, deadlock free, and b-bounded.

$N \oplus C$ is a closed net because of item (1) in Def. 11. Next, we show that $N \oplus C$ is b-bounded. Let m (m') be a marking of C (N) such that $m + m'$ is a reachable marking of $N \oplus C$ that violates the bound. Let v denote the trace of $env(C)$ that corresponds to the run from m_C to m. As ϱ exists, v is also a trace in $OG_b(N)$ and so it is in $env(N)$. By the construction of $OG_b(N)$, the corresponding markings in $env(N)$ do not violate the bound, so it suffices to assume that m violates the bound in $env(C)$. Then, $(m, Q) \in \varrho$ with $Q = Q_\emptyset$ by assumption. However, this implies that $m + m'$ is not reachable in $M_{N \oplus C}$, which is a contradiction to our assumption. Thus, $N \oplus C$ is b-bounded.

Finally, we show that $N \oplus C$ is deadlock free. Let m (m') be a marking of C (N) such that $m + m'$ is a reachable marking of $N \oplus C$. Marking m is also a state in $beh_b(C)$. From the existence of ϱ we conclude that there exists a node Q of $OG_b(N)$ with $(m, Q) \in \varrho$. Further, we have $m' \in Q$; otherwise, $N \oplus C$ is not b-bounded. Assume m is stable in $env(C)$ and m' is stable in $env(N)$; otherwise, $m + m'$ is no deadlock of $N \oplus C$ by Def. 4. Then $m \models \psi_1(m') \vee \psi_2(m')$ by assumption. If $m \models \psi_1(m')$, then there exists $x \in (I \uplus O)$ with $m \stackrel{x}{\rightarrow}$ in $beh_b(C)$ by Def. 11. The corresponding transition is also enabled in $N \oplus C$; thus, $m + m'$ is no deadlock. If $m \models \psi_2(m')$, then $m \in \Omega_{env(C)}$ by Def. 11 and $m' \in \Omega_{env(N)}$ by Def. 10. Thus, $m + m' \in \Omega_{N \oplus C}$ by Def. 3 and $m + m'$ is no deadlock of $N \oplus C$. □

The minimal simulation relation of $beh_b(C)$ by $OG_b(N)$ can be computed in time and space proportional to $O(|beh_b(C)| \cdot |OG_b(N)|)$. Together with the annotation check, matching $beh_b(C)$ with $OG_b(N)$ has a complexity of $O(|beh_b(C)| \cdot |OG_b(N)| \cdot 2^{k+1})$, whereas $k = |I \uplus O|$ denotes the size of the interface. Consequently, checking whether an open net is a b-controller is decidable.

Theorem 13 (decidability of b-controllability). *Checking whether an open net is a b-controller of another open net= is decidable for every $b \in \mathbb{N}^+$.*

4 Accordance

An algorithm to decide accordance for two open nets *Spec* and *Impl* must decide whether every controller of *Spec* is also a controller of *Impl*. As an open net has potentially infinitely many controllers, we must check inclusion of two infinite sets. Because the set of all controllers of an open net can be represented in a finite manner using the operating guideline, we may use the operating guidelines of *Spec* and *Impl* to decide that *Impl* accords with *Spec*.

The b-accordance relation has been defined by Stahl and Vogler [8] and they showed that it is a precongruence for composition operator $\oplus$ and therefore supports compositional reasoning.

Definition 14 (b-accordance). Let $b \in \mathbb{N}^+$. For interface equivalent open nets *Impl* and *Spec*, *Impl b-accords with Spec*, denoted by $Impl \sqsubseteq^b_{acc} Spec$, if for all open nets C hold: C is a b-controller of *Spec* implies C is a b-controller of *Impl*.

We show that deciding accordance of *Impl* and *Spec* reduces to checking that the operating guideline of *Spec* simulates the operating guideline of *Impl* and that the corresponding formulae of related states imply each other.

Definition 15 (b-refinement). Let $b \in \mathbb{N}^+$. For interface equivalent open nets *Impl* and *Spec*, $OG_b(Impl)$ *b-refines* $OG_b(Spec)$, denoted by $OG_b(Impl) \sqsubseteq^b_{ref} OG_b(Spec)$, if there exists a minimal simulation ϱ of $OG_b(Spec)$ by $OG_b(Impl)$ such that for each pair of nodes $(Q, Q') \in \varrho$:

1. $Q = Q_\emptyset$ implies $Q' = Q_\emptyset{}'$, and
2. the formula $\phi_{OG_b(Spec)}(Q) \Rightarrow \phi_{OG_b(Impl)}(Q')$ is a tautology.

The first item is crucial; otherwise, we could have a b-controller of *Spec* that is not a b-controller of *Impl* because it violates the bound only in the composition with *Impl* (the respective state is not reachable in the composition with *Spec*).

Consider Fig. 4. $OG_1(Impl)$ 1-refines $OG_1(Spec)$, but $OG_1(Spec)$ does not 1-refine $OG_1(Impl)$: Node $Q_\emptyset$ of $OG_1(Impl)$ is related with node Q_1 of $OG_1(Spec)$, thereby violating item (1) of Def. 15.

The next theorem justifies that refinement of operating guidelines and accordance coincide.

Theorem 16 (b-accordance vs. b-refinement). *Let $b \in \mathbb{N}^+$. For interface equivalent open nets Impl and Spec, $Impl \sqsubseteq^b_{acc} Spec$ iff $OG_b(Impl) \sqsubseteq^b_{ref} OG_b(Spec)$.*

Proof. Let $OG_b(Spec) = (\mathcal{Q}, E, Q_0, \Sigma_{in}, \Sigma_{out}, \phi)$ and $OG_b(Impl) = (\mathcal{Q}', E', Q_0', \Sigma_{in}, \Sigma_{out}, \phi')$ be the operating guidelines of open nets *Spec* and *Impl*, respectively.

($\Rightarrow$): Let $Impl \sqsubseteq^b_{acc} Spec$. Consider an open net C whose behavior $beh_b(C)$ is isomorph to the underlying automaton of $OG_b(Spec)$ and that has a final state if literal *final* occurs in the annotation of the respective node. Clearly, C is a

b-controller of *Spec* and of *Impl*. Thus, by Definition 11, there exists a minimal simulation relation of $beh_b(C)$ by $OG_b(Impl)$, and hence there is a minimal simulation relation ϱ of $OG_b(Spec)$ by $OG_b(Impl)$.

Let $Q \in \mathcal{Q}$, and let β be an arbitrary assignment to literals occurring in $\phi(Q)$ with β evaluates $\phi(Q)$ to *true*. Remove from the underlying automaton of $OG_b(Spec)$ and node Q all outgoing, x-labeled edges where $\beta(Q)(x)$ is *false*. By Definition 11, the corresponding automaton still matches with *Spec* and thus with *Impl*. Let $Q' \in \mathcal{Q}'$ with $(Q, Q') \in \varrho$. Using Definition 11 again, we can see that β satisfies $\phi'(Q')$ as well. Thus, $\phi(Q) \Rightarrow \phi'(Q')$ is a tautology, for all $(Q, Q') \in \varrho$.

Assume now that $Q = Q_\emptyset$. A b-controller C of *Spec* could be in a marking m that violates bound b, and m is related with $Q_\emptyset$. By assumption, C is a b-controller of *Impl* and hence we conclude that for all $Q' \in \mathcal{Q}'$, $(Q_\emptyset, Q')$ in the simulation relation of $OG_b(Spec)$ by $OG_b(Impl)$ implies $Q' = {Q_\emptyset}'$ (as otherwise $Impl \oplus C$ is not b-bounded).

($\Leftarrow$): Let $OG_b(Impl) \sqsubseteq^b_{ref} OG_b(Spec)$ and C be a b-controller of *Spec*. We have to show that C is b-controller of *Impl*, too.

By Definition 11, there exists a minimal simulation relation $\varrho_{beh_b(C), OG_b(Spec)}$ of $beh_b(C)$ by $OG_b(Spec)$ and, by assumption, we also have a minimal simulation relation $\varrho_{OG_b(Spec), OG_b(Impl)}$ of $OG_b(Spec)$ by $OG_b(Impl)$. As simulation is transitive we conclude that $\varrho_{beh_b(C), OG_b(Impl)}$ is a simulation relation of $beh_b(C)$ by $OG_b(Impl)$. Relation $\varrho_{beh_b(C), OG_b(Impl)}$ is even a minimal simulation relation, because the underlying automata of $OG_b(Spec)$ and $OG_b(Impl)$ are deterministic by construction.

By assumption, $beh_b(C)$ matches with $OG_b(Spec)$; that is, for all markings m with $(m, Q) \in \varrho_{beh_b(C), OG_b(Spec)}$ and m is stable in $env(C)$, m satisfies $\phi(Q)$. In addition, we know $\phi(Q) \Rightarrow \phi'(Q')$, for all $(Q, Q') \in \varrho_{OG_b(Spec), OG_b(Impl)}$. Hence, m satisfies $\phi(Q')$, for all $(m, Q') \in \varrho_{beh_b(C), OG_b(Impl)}$.

Suppose there exists a marking m of C that is not b-bounded. Then, by Definition 11, for all $Q \in \mathcal{Q}$, $(m, Q) \in \varrho_{beh_b(C), OG_b(Spec)}$ implies $Q = Q_\emptyset$. By assumption, for each pair of nodes $(Q, Q') \in \varrho_{OG_b(Spec), OG_b(Impl)}$, $Q = Q_\emptyset$ implies $Q' = {Q_\emptyset}'$; thus, we conclude $(m, Q') \in \varrho_{beh_b(C), OG_b(Impl)}$ implies $Q' = {Q_\emptyset}'$. □

We proceed with a short complexity analysis. Let $b \in \mathbb{N}^+$, and let *Impl* and *Spec* be interface equivalent open nets. A minimal simulation relation of $OG_b(Impl)$ by $OG_b(Spec)$ can be computed in time and space proportional to $O(|OG_b(Impl)| \cdot |OG_b(Spec)|)$. Let $k = |I \uplus O|$ denote the size of the interface. Then, checking whether *Impl* b-refines *Spec* has a complexity of $O(|OG_b(Impl)| \cdot |OG_b(Spec)| \cdot 2^{k+1})$. So checking b-accordance is decidable.

Theorem 17 (decidability of b-accordance). *Checking b-accordance of two open nets is decidable for every $b \in \mathbb{N}^+$.*

5 Conclusion

We have investigated the accordance precongruence of services. A service *Impl* accords with a service *Spec* if every controller of *Spec* (i.e., every service that

deadlock freely communicates with *Spec*) is also a controller of *Impl*. We have presented a novel way to decide accordance. To this end, we used the notion of an operating guideline [4], which represents all controllers of a service in a finite manner. We have adapted the procedure of checking whether a service is a controller of an a given service and is, thus, contained in the operating guideline. In addition, we have also adapted the procedure for deciding accordance [7] for two services *Spec* and *Impl* based on their operating guidelines.

In contrast to [4], we considered controllers with unbounded interior. This caused the adaptation of the techniques introduced in [4,7], because we need to distinguish whether a controller can potentially violate the bound in the composition or not. The definition of matching (see Def. 11) extends the respective definition in [4] by item 2(a), where we require that states, in which the controller violates the bound, are not reachable in the composition. Similar, item (1) in the definition of operating guideline refinement (see Def. 15) extends the respective definition in [7]. Also here, we assign a more prominent role to the empty node: The new accordance check has to distinguish whether an input is enabled in the empty node or in another *true* annotated node—that is, whether the input is enabled in a reachable state or not.

In ongoing work, we aim to study efficient procedures to decide accordance for stricter termination criteria than deadlock freedom, including responsiveness [10] (i.e., controllers either terminate or have the possibility to communicate) and weak termination (i.e., the service has always the possibility to terminate).

References

1. Kaschner, K.: Conformance testing for asynchronously communicating services. In: ICSOC 2011. LNCS, vol. 7084, pp. 108–124. Springer (2011)
2. Liske, N., Lohmann, N., Stahl, C., Wolf, K.: Another approach to service instance migration. In: ICSOC 2009. pp. 607–621. LNCS 5900, Springer-Verlag (2009)
3. Lohmann, N.: Correcting deadlocking service choreographies using a simulation-based graph edit distance. In: BPM 2008. pp. 132–147. LNCS 5240, Springer-Verlag (2008)
4. Lohmann, N., Massuthe, P., Wolf, K.: Operating guidelines for finite-state services. In: ICATPN 2007. LNCS, vol. 4546, pp. 321–341. Springer (2007)
5. Lohmann, N., Wolf, K.: Compact representations and efficient algorithms for operating guidelines. Fundam. Inform. 107, 1–19 (2011)
6. Papazoglou, M.P.: Web Services: Principles and Technology. Pearson (2007)
7. Stahl, C., Massuthe, P., Bretschneider, J.: Deciding substitutability of services with operating guidelines. In: ToPNoC II. pp. 172–191. LNCS 5460, Springer (2009)
8. Stahl, C., Vogler, W.: A trace-based service semantics guaranteeing deadlock freedom. Acta Informatica 49(2), 69–103 (2012)
9. Vogler, W.: Modular Construction and Partial Order Semantics of Petri Nets, LNCS, vol. 625. Springer (1992)
10. Vogler, W., Stahl, C., Müller, R.: A trace-based semantics for responsiveness. In: ACSD 2012. IEEE Computer Society (2012), to appear

Compositional analysis of modular Petri nets using hierarchical state space abstraction

Yves-Stan Le Cornec

IBISC, University of Évry, 23 bd de France, 91037 Évry, France
yves-stan.lecornec@ibisc.univ-evry.fr

Abstract. We propose an approach to perform efficient model-checking of μ-calculus formulae on modular Petri nets. Given a formula φ, each module can be analysed separately, possibly yielding a conclusion about the truth value of φ on the global system. When no conclusion can be drawn locally, a minimal state space preserving φ is computed for the module and can be incrementally composed with others, thus enabling for hierarchical analysis of a modular Petri net in a bottom-up fashion.

1 Introduction

State space explosion is a well known problem when dealing with model-checking of large systems. One way to address this problem in the context of Petri nets is *modularity*: a large Petri net is decomposed into subsystems which are then synchronised on shared places or transitions [2]. When only transitions are shared across sub-systems, like in [6], one way to alleviate state space explosion is to build a *modular state space* that consists of the state space of its subsystems (*i.e.*, its modules) and a synchronisation graph of them. This is usually a much smaller object than the state space of the full Petri net (*i.e.*, with all modules combined).

In this paper, we propose another way to analyse modular Petri nets when the goal is to model-check properties expressed as modal μ-calculus formulae, *i.e.*, to verify whether the state space of a modular Petri net is a model for a given formula φ. Our approach allows to analyse modules independently of each other. When the formula depends only on one module, only this particular module needs to be analysed. When the formula depends on several modules, each can be processed separately and its state space minimised before being combined with the others, *i.e.*, we compute a smaller transition system that is equivalent to the initial one with respect to the formula φ. This minimisation is an extension to the modal μ-calculus of the approach defined in [1] for CTL. Furthermore, our approach is fully compositional: on the one hand, the semantics of any composition of modules is equivalent to the synchronised product of the individual semantics of each module; on the other hand, minimisation of the semantics preserves the truth value of φ. So, a system composed of several modules can be decomposed into an arbitrary hierarchy forming a tree in which each leaf is a module and each internal node corresponds to the composition of the modules below it. Analysis can be performed by traversing this tree bottom-up in such a way that, at each node, we consider a particular subsystem whose

semantics can be computed and analysed so that, either we can raise some global conclusion about the truth of φ, or we can minimise the semantics (which will be reused at the upper level) while preserving the truth of φ. This approach leaves a lot of room to define strategies to choose an optimal order of analysis of modules in order to minimise the amount of work necessary to bring the conclusion. The current paper concentrates on defining the analysis method, this kind of optimisations being left to future work.

The rest of the paper is organised as follows. In the next section, we recall main definitions about modular Petri nets and define the semantics in terms of labelled transition systems. Next, we define the modal μ-calculus logic and its semantics. Section 4 forms the core of our contribution, defining the formula-dependent abstraction and giving the main results that enable hierarchical analysis and abstraction. For readability, proofs are moved in the appendix, after a conclusion section with perspectives.

2 Modular Petri nets

In the following, we consider place/transitions nets for simplicity, but a generalisation to high-level Petri nets (in particular to coloured Petri nets) is straightforward because our work is based on the labelled transitions systems used for the Petri nets semantics. To start with, let us recall the definition of Petri nets to fix the notations.

Definition 1. *A* Petri net $N \stackrel{\text{df}}{=} (P, T, W)$ *is a tuple such that:*

- P *is the finite set of* places*;*
- T *is the finite set of* transitions*, such that* $P \cap T = \emptyset$*;*
- W *is a multiset over* $(P \times T) \cup (T \times P)$ *defining the* arcs weights*;*

For $t \in T$*, we denote by* ${}^\bullet t$ *(resp.,* $t^\bullet$*) the multiset over* P *such that for all* $s \in P$*,* ${}^\bullet t(s) \stackrel{\text{df}}{=} W(s,t)$ *(resp.,* $t^\bullet(s) \stackrel{\text{df}}{=} W(t,s)$*).*

A marking M *of* N *is a multiset over* P *indicating how many tokens each place holds. A transition* t *is* enabled *at marking* M *iff* ${}^\bullet t \leq M$*, in which case the* firing *of* t *yields a new marking* $M' \stackrel{\text{df}}{=} M - {}^\bullet t + t^\bullet$*. This is denoted by* $M\ [t\rangle\ M'$*, moreover, we denote by* $[M\rangle$ *the smallest set containing* M *such that if* $M' \in [M\rangle$ *and* $M'\ [t\rangle\ M''$ *then* $M'' \in [M\rangle$*. We assume that* $[M\rangle$ *is finite.*

Our definition of modular Petri nets is adapted from [6]. We use here non-disjoint sets of transitions instead of explicit *transitions fusion sets* to define the transitions shared across modules. This especially means that we would have to make copies of a transition in order to model a choice between different synchronizations.

Definition 2. *A* modular Petri net *is a collection of* modules $(N_1, \ldots N_n)$ *where each* N_i *is a Petri net* (P_i, T_i, W_i)*, and such that the* P_i*'s are pairwise disjoint. Transitions that belong to only one* T_i *are called* local *while those shared among at least two* T_i*'s are called* fused*. Such a modular net is equivalent to a* flat Petri

net *obtained as the component-wise union of its modules. Because this union is commutative and associative, we shall use a binary notation for it:* $N_1 \oplus \cdots \oplus N_n$.

Example 1. Figure 1 shows two modules which are part of a modular Petri net. Transition f_3 is assumed to be fused with another module not shown here. ⋄

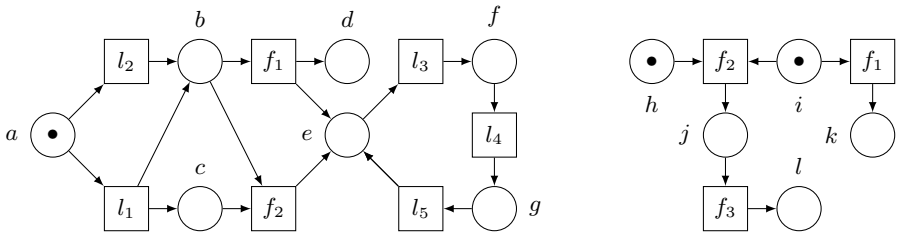


Fig. 1. Two modules part of a modular Petri net.

The semantics of Petri nets and modular Petri nets can be defined in terms of *labelled transitions systems* (*LTS*).

2.1 LTS semantics of Petri nets and modular Petri nets

A LTS is a tuple $S \stackrel{\text{df}}{=} (Q, q_0, A, R, L)$ where Q is a set of *states*, $q_0 \in Q$ is the *initial state*, A is the set of *actions* used as transition labels, $R \subseteq Q \times A \times Q$ is the set of *transitions* and L is a labelling of states with Boolean formulae on propositional variables from a set $\mathbb{V}$. A transition $(q, a, q') \in R$ is usually denoted by $q \xrightarrow{a} q'$.

Definition 3. *The* LTS semantics of a Petri net $N \stackrel{\text{df}}{=} (P, T, W)$ *initially marked by* M_0 *is the LTS* $[\![N]\!] \stackrel{\text{df}}{=} (Q, q_0, A, R, L)$ *such that:* $Q \stackrel{\text{df}}{=} [M_0\rangle$; $q_0 \stackrel{\text{df}}{=} M_0$; $A \stackrel{\text{df}}{=} T$; $R \stackrel{\text{df}}{=} \{M \xrightarrow{t} M' \mid M\ [t\rangle\ M'\}$; *and* $L(M) \stackrel{\text{df}}{=} \bigwedge_{M(p)>0} \texttt{p=M(p)}$ *with* $\mathbb{V} \stackrel{\text{df}}{=} \{\texttt{p=k} | p \in P, k \in \mathbb{N}^+\}$.

In this definition, states are labelled by a conjunction of propositional variables of the form `s=k` denoting the number of tokens in each non void place. This choice is arbitrary and can be changed in many ways, this will not affect the current work as long as we are able to evaluate atomic formulae. In order to bring modularity at the semantics level, we shall partition A as $A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$ corresponding respectively to the local and fused transitions of a modular Petri net.

Example 2. Figure 2 shows the LTS semantics of the modules from example 1.⋄

Definition 4. *The* modular LTS semantics of a modular Petri net $(N_i)_{1 \le i \le n}$, *where* $N_i \stackrel{\text{df}}{=} (P_i, T_i, W_i)$ *for all* i, *is a collection of LTS* $(Q_i, q_{0i}, A_i, R_i, L_i)_{1 \le i \le n}$ *where each* A_i *is partitioned as* $A_i^{\mathsf{loc}} \uplus A_i^{\mathsf{fus}}$ *such that, for* $1 \le i \le n$:

- $(Q_i, q_{0i}, A_i, R_i, L_i) = [\![N_i]\!]$ *is the LTS semantics of* N_i *considered alone;*
- $A_i^{\mathsf{loc}} \stackrel{\text{df}}{=} R_i \setminus \bigcup_{j \neq i} R_j$ *and* $A_i^{\mathsf{fus}} \stackrel{\text{df}}{=} A_i \setminus A_i^{\mathsf{loc}}$.

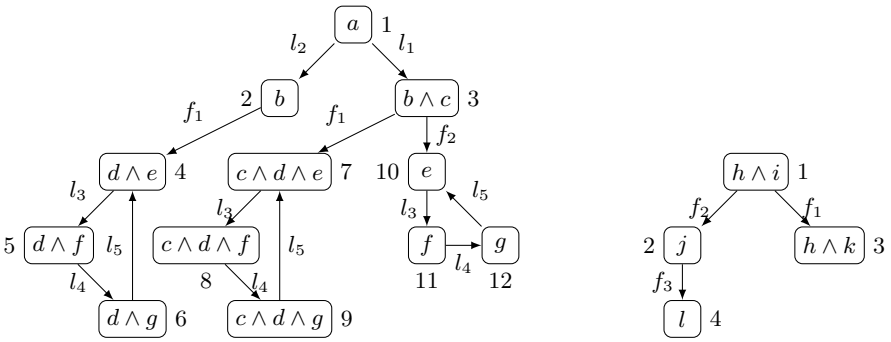


Fig. 2. LTS semantics of the modules from figure 1. For the left LTS, we have $A^{\mathsf{loc}} \stackrel{\text{df}}{=} \{l_1, l_2, l_3, l_4\}$ and $A^{\mathsf{fus}} \stackrel{\text{df}}{=} \{f_1, f_2\}$; for the right LTS we have $A^{\mathsf{loc}} \stackrel{\text{df}}{=} \emptyset$ and $A^{\mathsf{fus}} \stackrel{\text{df}}{=} \{f_1, f_2, f_3\}$. As there is at most one token per place, we write a instead of $a = 1$.

The collection of LTS obtained from a modular Petri net can be transformed into a single LTS by taking the *synchronised product* of its components, where synchronisation takes place on the fused transitions, which is the usual definition of a n-ary synchronised product. We denote by $x[i]$ the i-th component of a tuple x and by $x[i \leftarrow y_i]$ the tuple x in which the i-th component has been replaced by y_i, this latter notation is naturally extended to the replacement of several components.

Definition 5. *Let $(S_i)_{1 \leq i \leq n}$ be the LTS semantics of a modular Petri net with $S_i \stackrel{\text{df}}{=} (Q_i, q_{0i}, A_i, R_i, L_i)$ and $A_i \stackrel{\text{df}}{=} A_i^{\mathsf{loc}} \uplus A_i^{\mathsf{fus}}$, the* synchronised product *of the S_i's, is the LTS (Q, q_0, A, R, L) defined by:*

- $Q \stackrel{\text{df}}{=} \prod_{1 \leq i \leq n} Q_i$;
- $q_0 \stackrel{\text{df}}{=} (q_{01}, \ldots, q_{0n})$;
- $A \stackrel{\text{df}}{=} \bigcup_{1 \leq i \leq n} A_i$;
- *R is the smallest subset of $Q \times A \times Q$ such that $x \xrightarrow{a} y \in R$ iff either*
  - *it exists i such that $a \in A_i^{\mathsf{loc}}$, $x[i] \xrightarrow{a} y_i \in R_i$ and $y = x[i \leftarrow y_i]$,*
  - *or, for all i such that $a \in A_i^{\mathsf{fus}}$, we have $x[i] \xrightarrow{a} y_i \in R_i$ and $y = x[i \leftarrow y_i]$.*
- *for all $x \in Q$, $L(x) \stackrel{\text{df}}{=} \bigwedge_{1 \leq i \leq n} L_i(x[i])$.*

Because this product is associative and commutative, we shall also use a binary notation for it: $S_1 \otimes \cdots \otimes S_n$.

In the definition of R above, the first point corresponds to the cases where a module evolves on a local transition. So only one component of the compound state evolves. The second point corresponds to the firing of a fused transition, in which case all the modules sharing this transition must simultaneously fire and the corresponding components of the compound state will simultaneously evolve. Notice that, by definition of a fused transition, if $a \in A_i^{\mathsf{fus}}$ for some i,

then there exists at least one $j \neq i$ such that $a \in A_j^{\mathsf{fus}}$ also (otherwise, we would have $a \in A_i^{\mathsf{loc}}$).

From the definitions above, it immediately follows that the synchronised product of the LTS semantics of a modular Petri net is equivalent to the LTS semantics of the flat Petri net.

Theorem 1. *Let $(N_i)_{1 \leq i \leq n}$ be a modular Petri net. We have*

$$[\![N_1 \oplus \cdots \oplus N_n]\!] \quad \sim \quad [\![N_1]\!] \otimes \cdots \otimes [\![N_n]\!]$$

where $\sim$ denotes the isomorphism of LTS. □

Because of this, we can define the notation $[\![N_1, \ldots, N_n]\!] \stackrel{\text{df}}{=} [\![N_1]\!] \otimes \cdots \otimes [\![N_n]\!]$. These notations are intended to put into light a first level of compositionality. For example, consider a modular Petri net $(N_1, \ldots, N_5)$. It is possible to see it as, *e.g.*, three subsystems (N_1, N_2), (N_3, N_4) and N_5 and to compute $[\![N_1, N_2]\!] \otimes [\![N_3, N_4]\!] \otimes [\![N_5]\!]$, just like if we would have considered $(N_1 \oplus N_2, N_3 \oplus N_4, N_5)$ as the initial system, which is also equivalent to $(N_1 \oplus N_2) \oplus (N_3 \oplus N_4) \oplus (N_5)$. So we can decompose a modular Petri net into a hierarchy and compute the semantics at any level of this hierarchy. This is the first step towards full compositionality; the next step will be to introduce LTS minimisation with respect to a μ-calculus formula in order to be able to apply minimisation hierarchically.

3 The modal μ-calculus

The modal μ-calculus (or simply μ-calculus) is a temporal logic that encompasses widely used logics such as, in particular, CTL* (and thus also LTL and CTL) [5]. A μ-formula is derived from the following grammar, where B is a Boolean formula, X is a propositional variable and α is a set of actions:

$$\varphi \quad ::= \quad B \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\alpha\rangle\varphi \mid \mu X.\varphi \mid X$$

Moreover, in a formula $\mu X.\varphi$, φ must be positive in the variable X, *i.e.*, every free occurrence of X must be in the scope of an even number of negations $\neg$.

A formula φ is evaluated over a LTS $S \stackrel{\text{df}}{=} (Q, A, R, L)$ and can be seen as a function of its free variables to 2^Q. In particular, if φ is a closed formula then it is a function with no arguments that returns the subset of Q where φ holds. In formula $\mu X.\langle a\rangle X \vee B$, the sub-formula $\langle a\rangle X \vee B$ defines a function that, given $X \subseteq Q$, returns the states y such that either $y \xrightarrow{a} x$ for some $x \in X$, or B holds on y. From this point of view, $\mu X.\varphi$ is the least fixed point of function φ. More generally, the semantics φ^N of φ over S is defined as follows:

- B holds in every state whose label implies B: $B^S \stackrel{\text{df}}{=} \{x \in Q \mid L(x) \Rightarrow B\}$;
- $\varphi_1 \vee \varphi_2$ holds in every state where φ_1 or φ_2 holds: $(\varphi_1 \vee \varphi_2)^S \stackrel{\text{df}}{=} \varphi_1^S \cup \varphi_2^S$;
- $\neg\varphi$ holds in every state where φ does not: $(\neg\varphi)^S \stackrel{\text{df}}{=} Q \setminus \varphi^S$;
- $\langle\alpha\rangle\varphi$ holds in every state where a transition labelled by $a \in \alpha$ leads to a state where φ holds: $(\langle\alpha\rangle\varphi)^S \stackrel{\text{df}}{=} \{x \in Q \mid \exists y \in \varphi^S, \exists a \in \alpha, x \xrightarrow{a} y \in R\}$;

– $\mu X.\varphi$ is the least fixed point of function φ: $(\mu X.\varphi)^S \stackrel{\text{df}}{=} \bigcap_{\rho \in 2^Q \wedge \varphi(\rho) \subseteq \rho} \rho$;
– $X^S \stackrel{\text{df}}{=} X$, can be seen as the identity function.

For closed formulae, φ^S is a subset of Q, which can be equivalently seen as the image of a function with no arguments. But for formulae with free variables, φ^S is a function exactly like φ is and the above definition can be read as transformation of functions. For instance, for a φ with a single free variable X, the definition of $(\neg\varphi)^S$ can be reformulated as $(\neg\varphi)^S(X) \stackrel{\text{df}}{=} Q \setminus \varphi^S(X)$. Note that, to simplify the definitions in the sequel, we have considered α as a set of actions in $\langle\alpha\rangle$ instead of as a single action as more usual. Moreover, because the semantics of a formula is a set of states, we may use one or the other form interchangeably.

The effective construction of $\mu X.\varphi$ is made inductively by defining $0X.\varphi \stackrel{\text{df}}{=} \emptyset$, and $nX.\varphi \stackrel{\text{df}}{=} \varphi[X \leftarrow (n-1)X.\varphi]$ where $\varphi[X \leftarrow Y]$ denotes φ in which variable X is substituted by Y everywhere. Knaster-Tarski's theorem ensures that there exists $k \in \mathbb{N}$ such that $kX.\varphi = \mu X.\varphi$. Indeed, φ is a monotonous function over a complete lattice, and thus the set of its fixed-points is also a complete lattice [8].

Example 3. Let us consider the LTS from the left of figure 2 and the formula $\mu X.(\langle A\rangle X \vee d)$ (which means that the module can reach a state where place d is marked). We can compute the least fixed-point of $\varphi \stackrel{\text{df}}{=} \langle A\rangle X \vee d$ as follows:

– $0X.\varphi \stackrel{\text{df}}{=} \emptyset$
– $1X.\varphi \stackrel{\text{df}}{=} \varphi(0X.\varphi) = (\langle A\rangle X)(\emptyset) \cup d() = \emptyset \cup \{4,5,6,7,8,9\}$
– $2X.\varphi \stackrel{\text{df}}{=} (\langle A\rangle X)(1X.\varphi) \cup \{4,5,6,7,8,9\} = \{2,3,4,5,6,7,8,9\}$
– $3X.\varphi \stackrel{\text{df}}{=} (\langle A\rangle X)(2X.\varphi) \cup \{4,5,6,7,8,9\} = \{1,2,3,4,5,6,7,8,9\}$
– $4X.\varphi \stackrel{\text{df}}{=} (\langle A\rangle X)(3X.\varphi) \cup \{4,5,6,7,8,9\} = 3X.\varphi$ ◇

To write formulae more comfortably, we can use the following operators:

– $\varphi_1 \wedge \varphi_2 \stackrel{\text{df}}{=} \neg(\neg\varphi_1 \vee \neg\varphi_2)$;
– $[\alpha]\varphi \stackrel{\text{df}}{=} \neg\langle\alpha\rangle\neg\varphi$, which yields $([\alpha]\varphi)^S \stackrel{\text{df}}{=} \{x \in Q \mid \forall y \in \varphi^S, x \xrightarrow{a} y \in T\}$;
– $\nu X.\varphi \stackrel{\text{df}}{=} \neg\mu X.\neg\varphi[X \leftarrow \neg X]$ is the greatest fixed-point of φ.

Finally, for $q \in Q$, we write $S, q \models \varphi$ iff $q \in \varphi^S$, and $S \models \varphi$ iff $S, q_0 \models \varphi$.

4 Formula-dependent abstraction

In this section, we introduce an operation $\lfloor S \rfloor_\varphi$ that, given the LTS S of a module and a formula φ, returns a LTS, (usually) smaller that S without changing the truth of φ, neither over S, nor over the global system (*i.e.*, S synchronised with the LTS of the other modules). To do so, we define an equivalence relation (initially between states of a same LTS, then extended to LTS), denoted by $\sim^\varphi$, that preserves the truth value of φ on the global system and is a congruence with respect to synchronised product $\otimes$.

Let $S = (q_0, Q, A, R, L)$ with $A \stackrel{\text{df}}{=} A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$ be the LTS of a module, $q \in Q$ a state, and φ a formula. Let also Ex be the actions appearing in φ but not in A, which corresponds to the context of S, *i.e.*, the other modules. To start with,

we define the set Pass^{φ} such that if $q \in \mathsf{Pass}^{\varphi}$ then φ is necessarily true on any bigger system in which S is a module in state q. Similarly, we define Fail^{φ} such that if $q \in \mathsf{Fail}^{\varphi}$ then φ is necessarily false on any bigger system encompassing S in state q. We shall write $\mathsf{Pass}^{\varphi}(S)$ or $\mathsf{Fail}^{\varphi}(S)$ when the LTS of interest needs to be precised. We say that formula φ can be evaluated on a state q belonging to S if $q \in \mathsf{Pass}^{\varphi} \cup \mathsf{Fail}^{\varphi}$, which is denoted by $\varphi \stackrel{?}{=\!|} S, q$. This means that we can conclude about the truth of φ in state q independently of the context in which S may be embedded as a module. Similarly, φ can be evaluated on S if it can be evaluated it on its initial state, which is denoted by $\varphi \stackrel{?}{=\!|} S$.

The definition below is made with respect to a context Σ that is a map from the free variables of a formula to sets of states (when needed, Σ may be seen as a set of pairs). For a formula φ that contains free variables $X_1, \ldots, X_n$ that do not appear in the environment Σ, $\mathsf{Pass}^{\Sigma,\varphi}$ is the function $(x_1, \ldots, x_n) \mapsto \mathsf{Pass}^{\Sigma \cup \{(X_i, x_i) | 1 \leq i \leq n\}, \varphi}$.

Definition 6. *Let φ be a formula and $S \stackrel{\text{df}}{=} (q_0, Q, A, R, L)$ with $A \stackrel{\text{df}}{=} A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$ be a LTS. We set $\mathsf{Pass}^{\varphi} \stackrel{\text{df}}{=} \mathsf{Pass}^{\emptyset,\varphi}$ and $\mathsf{Fail}^{\varphi} \stackrel{\text{df}}{=} \mathsf{Fail}^{\emptyset,\varphi}$, where $\mathsf{Pass}^{\emptyset,\varphi}$ and $\mathsf{Fail}^{\emptyset,\varphi}$ are functions defined inductively on the syntax of φ:*

- $\mathsf{Pass}^{\Sigma,X} \stackrel{\text{df}}{=} \Sigma(X)$;
- $\mathsf{Pass}^{\Sigma,B} \stackrel{\text{df}}{=} \{x \in Q \mid L(x) \Rightarrow B\}$;
- $\mathsf{Pass}^{\Sigma,\neg\varphi} \stackrel{\text{df}}{=} \mathsf{Fail}^{\Sigma,\varphi}$;
- $\mathsf{Pass}^{\Sigma,\varphi_1 \vee \varphi_2} \stackrel{\text{df}}{=} \mathsf{Pass}^{\Sigma,\varphi_1} \cup \mathsf{Pass}^{\Sigma,\varphi_2}$;
- $\mathsf{Pass}^{\Sigma,\langle\alpha\rangle\varphi} \stackrel{\text{df}}{=} (\langle \alpha \cap A^{\mathsf{loc}} \rangle X)(\mathsf{Pass}^{\Sigma,\varphi})$;
- $\mathsf{Pass}^{\Sigma,\mu X.\varphi} \stackrel{\text{df}}{=} \mu X.\mathsf{Pass}^{\varphi}$.
- $\mathsf{Fail}^{\Sigma,X} \stackrel{\text{df}}{=} \Sigma(X)$;
- $\mathsf{Fail}^{\Sigma,B} \stackrel{\text{df}}{=} \mathsf{Pass}^{\Sigma,\neg B}$;
- $\mathsf{Fail}^{\Sigma,\neg\varphi} \stackrel{\text{df}}{=} \mathsf{Pass}^{\Sigma,\varphi}$;
- $\mathsf{Fail}^{\Sigma,\varphi_1 \vee \varphi_2} \stackrel{\text{df}}{=} \mathsf{Fail}^{\Sigma,\varphi_1} \cap \mathsf{Fail}^{\Sigma,\varphi_2}$;
- $\mathsf{Fail}^{\Sigma,\langle\alpha\rangle\varphi} \stackrel{\text{df}}{=} ([\alpha \setminus Ex]X)(\mathsf{Fail}^{\Sigma,\varphi}) \cap F$, *where* $F \stackrel{\text{df}}{=} Q$ *if* $A \cap Ex = \emptyset$ *and* $F \stackrel{\text{df}}{=} \mathsf{Fail}^{\Sigma,\varphi}$ *otherwise;*
- $\mathsf{Fail}^{\Sigma,\mu X.\varphi} \stackrel{\text{df}}{=} \nu X.\mathsf{Fail}^{\Sigma,\varphi}$.

Together with Pass^{φ} and Fail^{φ}, we aim to define $\sim^{\varphi}$ as a relation between the states such that if $x \sim^{\varphi} y$, then, in a larger system embedding S, formula φ does not allow to distinguish the states embedding x or y. (Thus it will be possible to reduce S by merging these two states.) In order to compute relation $\sim^{\varphi}$ we define $\mathfrak{F}^{\Sigma,\varphi}$ and build $\mathfrak{F}^{\varphi} \stackrel{\text{df}}{=} \mathfrak{F}^{\emptyset,\varphi}$.

The computation of $\mathfrak{F}^{\emptyset,\varphi}$ described in definition 7, yields a triple (p, f, r) such that at the end of computation, $p = \mathsf{Pass}^{\varphi}$,$f = \mathsf{Fail}^{\varphi}$ and $r \stackrel{\text{df}}{=} \sim^{\varphi}$ (this is not necessarily the case at every step). The rules used to build $\mathfrak{F}^{\emptyset,\varphi}$, as shown in exemple 4, operate on elements from $Q \times Q \times Q^2$. As in the definitions of Pass and Fail, if φ contains free variables $X_1, \ldots, X_n$ which do not appear in the environment Σ, then $\mathfrak{F}^{\Sigma,\varphi}$ is the function $(x_1, \ldots, x_n) \mapsto \mathfrak{F}^{\Sigma \cup \{(X_i, x_i) | 1 \leq i \leq n\}, \varphi}$. However unlike in the previous definition, environment Σ now takes values from $Q \times Q \times Q^2$. We also define Σ^p and Σ^f as the projections of Σ on its first and second components, *i.e.*, the smallest environments such that if $(X, (p, f, r)) \in \Sigma$, then $(X, p) \in \Sigma^p$ and $(X, f) \in \Sigma^f$.

Definition 7. *Let φ be a formula and $S \stackrel{\text{df}}{=} (Q, A, R, L)$ with $A \stackrel{\text{df}}{=} A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$ be a LTS. $\mathfrak{F}^{\Sigma,\varphi}$ is defined recursively on the syntax of φ as follows:*

1. $\mathfrak{F}^{\Sigma,B} \stackrel{\text{df}}{=} (\mathsf{Pass}^{\Sigma,B}, \mathsf{Fail}^{\Sigma,B}, \{(x,y) \mid (L(x) \Rightarrow B) \Leftrightarrow (L(y) \Rightarrow B)\})$.
2. $\mathfrak{F}^{\Sigma,\neg\varphi_1} \stackrel{\text{df}}{=} \mathfrak{F}^{\Sigma,\neg}(\mathfrak{F}^{\Sigma,\varphi_1})$
   with $\mathfrak{F}^{\Sigma,\neg}(p,f,r) \stackrel{\text{df}}{=} (f,p,r)$.
3. $\mathfrak{F}^{\Sigma,\varphi_1\vee\varphi_2} \stackrel{\text{df}}{=} \mathfrak{F}^{\Sigma,\vee}(\mathfrak{F}^{\Sigma,\varphi_1}, \mathfrak{F}^{\Sigma,\varphi_2})$
   with $\mathfrak{F}^{\Sigma,\vee}((p_1,f_1,r_1),(f_2,p_2,r_2)) \stackrel{\text{df}}{=} (p_1 \cup p_2, f_1 \cap f_2, r_1 \cap r_2)$.
4. $\mathfrak{F}^{\Sigma,\langle\alpha\rangle\varphi_1} \stackrel{\text{df}}{=} \mathfrak{F}^{\Sigma,\langle\alpha\rangle}(\mathfrak{F}^{\Sigma,\varphi_1})$
   with $\mathfrak{F}^{\Sigma,\langle\alpha\rangle}(p,f,r) \stackrel{\text{df}}{=} (\mathsf{Pass}^{\Sigma^p,\langle\alpha\rangle X}(p), \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle X}(f), r')$ *where* $(x,y) \in r'$ *iff* $(x,y) \in r$ *and either*
   (a) $x \in \mathsf{Pass}^{\Sigma^p,\langle\alpha\rangle X}(p)$ *and* $y \in \mathsf{Pass}^{\Sigma^p,\langle\alpha\rangle X}(p)$,
   (b) *or,* $x \in \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle X}(f)$ *and* $y \in \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle X}(f)$,
   (c) *or, we have*
      i. *for every* $a \in \alpha \cap A^{\mathsf{fus}}$, *if* $x \xrightarrow{a} x' \in R$ *and* $x' \notin f$ *then it exists* $y \xrightarrow{a} y' \in R$ *such that* $(x',y') \in r$,
      ii. *and, for every* $a \in \alpha \cap A^{\mathsf{fus}}$, *if* $y \xrightarrow{a} y' \in R$ *and* $y' \notin f$ *then it exists* $x \xrightarrow{a} x' \in R$ *such that* $(x',y') \in r$,
      iii. *and, for every* $a \in \alpha \cap A^{\mathsf{loc}}$, *if* $x \xrightarrow{a} x' \in R$ *and* $x' \notin f$ *then it exists* $y \xrightarrow{a'} y' \in R$ *such that* $a' \in \alpha \cap A^{\mathsf{loc}}$ *and* $(x',y') \in r$,
      iv. *and, for every* $a \in \alpha \cap A^{\mathsf{loc}}$, *if* $y \xrightarrow{a} y' \in R$ *and* $x' \notin f$ *then it exists* $x \xrightarrow{a'} x' \in R$ *such that* $a' \in \alpha \cap A^{\mathsf{loc}}$ *and* $(x',y') \in r$.
5. $\mathfrak{F}^{\Sigma,X} \stackrel{\text{df}}{=} \Sigma(X)$.
6. $\mathfrak{F}^{\Sigma,\mu X.\varphi_1}$ *is the fixed-point reached by iterating function* $\mathfrak{F}^{\Sigma,\varphi_1}$ *starting from* $(Pass^{\Sigma^p,\mu X.\varphi_1}, Fail^{\Sigma^f,\mu X.\varphi_1}, r_0)$ *with* $(x,y) \in r_0$ *iff either*
   (a) $x \in Pass^{\Sigma^p,\mu X.\varphi_1}$ *and* $y \in Pass^{\Sigma^p,\mu X.\varphi_1}$,
   (b) *or* $x \in Fail^{\Sigma^f,\mu X.\varphi_1}$ *and* $y \in Fail^{\Sigma^f,\mu X.\varphi_1}$,
   (c) *or* $x, y \in Q \setminus (Fail^{\Sigma^f,\mu X.\varphi_1} \cup Pass^{\Sigma^p,\mu X.\varphi_1})$.

With regard to $\sim^{\varphi}$, this definition can be intuitively understood as follows:

1. The labels of two equivalent states must be identical with respect to the atomic formulae which appear in B.
2. The relations corresponding to a formula and to its negation are the same.
3. Two equivalent states must be equivalent on both sub-formulae.
4. When φ involves the next states through $\langle\alpha\rangle$, two equivalent states x and y must be equivalent w.r.t. the sub-formula and either
   (a) both x and y ensure that φ globally holds,
   (b) or, both x and y ensure that φ does not hold globally,
   (c) or,
      i. if from x we can reach x' through a fused transition, then this must be possible from y through the same action, reaching a state y' equivalent to x'.Note that we only have to consider the x' which are not in the Fail set of the sub-formula,
      ii. and the same thing symmetrically for y.

iii. moreover, if from x we can reach x' (which does not belong to the Fail set of the sub-formula) through a local action, then this must be possible from y through the same or another local action from the set α, reaching a state y' equivalent to x',
iv. and the same thing symmetrically for y.

5. The value of X is fetched from the environment (by construction, we are always have every free variable in the environment).
6. Function $\mathfrak{F}^{\Sigma,\varphi_1}$ is repeatedly applied starting from the greatest relation for which $\mathsf{Pass}^{\Sigma,\mu X.\varphi_1}$ and $\mathsf{Fail}^{\Sigma,\mu X.\varphi_1}$ are equivalence classes. Each iteration (which is a composition of the previous rules) will then differentiate some states until we reach a fixed-point. This necessarily occurs because, for any relation r, we have $\mathfrak{F}^{\Sigma,\varphi_1}(\mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}, \mathsf{Fail}^{\Sigma^p,\mu X.\varphi_1}, r) = (\mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}, \mathsf{Fail}^{\Sigma^p,\mu X.\varphi_1}, r')$ with $r' \subseteq r$. So we always eventually reach a fixed-point when applying $\mathfrak{F}^{\Sigma,\varphi_1}$ repeatedly while computing $\mathfrak{F}^{\Sigma,\mu X.\varphi_1}$.

Definition 8. *Let φ be a closed formula, $\sim^{\varphi}$ is defined as the third component returned by $\mathfrak{F}^{\varphi}$.*

Example 4. Let $\varphi \stackrel{\text{df}}{=} B_1 \vee \langle\alpha\rangle B_2$ for some Boolean formulae B_1 and B_2. As defined above, we can compute

$$\mathfrak{F}^{B_1 \vee \langle\alpha\rangle B_2} = \mathfrak{F}^{\emptyset,\vee}(\mathfrak{F}^{\emptyset,B_1}, \mathfrak{F}^{\emptyset,\langle\alpha\rangle}(\mathfrak{F}^{\emptyset,B_2})) = (\mathsf{Pass}^{\varphi}, \mathsf{Fail}^{\varphi}, \sim^{\varphi}) \qquad \diamond$$

The next lemma states that we have indeed defined an equivalence relation. This equivalence is defined within the context of a single LTS, this can be generalised to compare two LTS.

Lemma 1. *Relation $\sim^{\varphi}$ as defined above is indeed an equivalence relation.*

Definition 9. *Let φ be a formula, and S_1 and S_2 be two LTS whose initial states are $q_{0,1}$ and $q_{0,2}$ respectively. S_1 is equivalent to S_2 w.r.t. φ, which is denoted by $S_1 \sim^{\varphi} S_2$, iff $q_{0,1} \sim^{\varphi} q_{0,2}$ in S defined as the component-wise disjoint union of S_1 and S_2.*

Using relation $\sim^{\varphi}$ we define the reduction of a LTS by merging its equivalent states, which is done by considering the quotient set of Q by $\sim^{\varphi}$.

Definition 10. *Let $S \stackrel{\text{df}}{=} (Q, q_0, A, R, L)$ be a LTS with $A \stackrel{\text{df}}{=} A^{\mathsf{loc}} \uplus A^{\mathsf{fus}}$, and φ be a formula. The* reduction of S w.r.t. φ, *denoted by $\lfloor S \rfloor_{\varphi}$, is the LTS (Q', q'_0, A', R', L') such that:*

- $Q' \stackrel{\text{df}}{=} Q/\!\sim^{\varphi}$ *is the quotient set of Q by $\sim^{\varphi}$;*
- $q'_0 \stackrel{\text{df}}{=} [q_0]_{\varphi}$ *is the equivalence class containing q_0;*
- A' *is exactly A and is partitioned the same way;*
- $R' \stackrel{\text{df}}{=} \{(c_1, a, c_2) \in Q' \times A' \times Q' \mid \exists q_1 \in c_1, \exists q_2 \in c_2, (q_1, a, q_2) \in R\}$;
- $L'(c) \stackrel{\text{df}}{=} \bigvee_{q \in [c]_{\varphi}} L(q)$.

We now introduce several properties of the definitions above, progressively leading to our main compositionality result. First, we state that Pass and Fail effectively allow to locally conclude about the global truth value of a formula.

Lemma 2. *Let* $S \stackrel{\text{df}}{=} S_1 \otimes \cdots \otimes S_n$ *be a product LTS and* $x \stackrel{\text{df}}{=} (x_1, \ldots, x_n)$ *one of its states.*

1. *If* $x_i \in \mathsf{Pass}^{\varphi}(S_i)$ *for some* $1 \leq i \leq n$*, then* $x \in \mathsf{Pass}^{\varphi}(S)$.
2. *If* $x_i \in \mathsf{Fail}^{\varphi}(S_i)$ *for some* $1 \leq i \leq n$*, then* $x \in \mathsf{Fail}^{\varphi}(S)$.

Then, we state that $\sim^{\varphi}$ correctly captures the states that are equivalent w.r.t. the capability to evaluate locally the global truth of φ. Moreover, it also preserves the truth value of φ.

Theorem 2. *Let* S *be a LTS,* φ *a formula, and* x *and* y *two states of* S *such that* $x \sim^{\varphi} y$*. If* $\varphi \stackrel{?}{=\!|} S, x$*, then* $\varphi \stackrel{?}{=\!|} S, y$ *and* $S, x \models \varphi \Leftrightarrow S, y \models \varphi$.

Corollary 1. *Let* φ *be a formula, and* S_1 *and* S_2 *two LTS such that* $S_1 \sim^{\varphi} S_2$*. If* $\varphi \stackrel{?}{=\!|} S_1$ *then* $\varphi \stackrel{?}{=\!|} S_2$ *and* $S_1 \models \varphi \Leftrightarrow S_2 \models \varphi$.

Moreover, the reduction w.r.t φ yields a LTS that is equivalent to the original one. Then, we state the consistency of reduction $\lfloor S \rfloor_{\varphi}$ w.r.t. the synchronised product, which allows to extend the previous lemma to compound LTS. Finally, this reduction is a congruence for the product of LTS, which means that we can replace any LTS of a product by the corresponding reduced LTS while preserving the equivalence relation.

Lemma 3. *Let* S *be a LTS and* φ *a formula, we have:* $S \sim^{\varphi} \lfloor S \rfloor_{\varphi}$.

Theorem 3. *Let* $S \stackrel{\text{df}}{=} S_1 \otimes \cdots \otimes S_n$ *be a product LTS,* $x \stackrel{\text{df}}{=} (x_1, \ldots, x_n)$ *and* $y \stackrel{\text{df}}{=} (y_1, \ldots, y_n)$ *two of its states, and* φ *a formula. If* $x_i \sim^{\varphi} y_i$ *for all* $1 \leq i \leq n$ *then* $x \sim^{\varphi} y$.

Corollary 2. $S_1 \otimes \cdots \otimes S_n \sim^{\varphi} \lfloor S_1 \rfloor_{\varphi} \otimes \cdots \otimes \lfloor S_n \rfloor_{\varphi}$

Combining these results with theorem 1, we can perform modular analysis with hierarchical reductions. Indeed, given a formula φ and a modular Petri net $(N_1, \ldots, N_n)$, let us define $\llbracket N_i \rrbracket_{\varphi} \stackrel{\text{df}}{=} \lfloor \llbracket N_i \rrbracket \rfloor_{\varphi}$, we have:

$$\llbracket N_1 \oplus \cdots \oplus N_n \rrbracket \quad \sim \quad \llbracket N_1 \rrbracket \otimes \cdots \otimes \llbracket N_n \rrbracket \quad \sim^{\varphi} \quad \llbracket N_1 \rrbracket_{\varphi} \otimes \cdots \otimes \llbracket N_n \rrbracket_{\varphi}$$

Furthermore, lemma 2 tells us that we can stop building the system as soon as we find a sub-system on which the formula can be evaluated, *i.e.*, as soon as $\varphi \stackrel{?}{=\!|} \llbracket N_i \rrbracket_{\varphi}$ for some i, because the truth value of that formula over the global system will be the same as over this sub-system. Finally, because modules can be freely associated and commuted, we can conduct the analysis hierarchically, reducing at each level and possibly stopping before the whole system semantics is constructed.

Example 5. Let us consider again the example 2 (and figure 2) and check that we can reach a state where the property $h \wedge d$ is true, that is expressed in μ-calculus as $\mu X.(\langle A \rangle X \vee (h \wedge d))$. Remember that we assumed there exists a third module synchronised over f_3. We will see that in this case, analysing the first two modules is sufficient to prove the property.

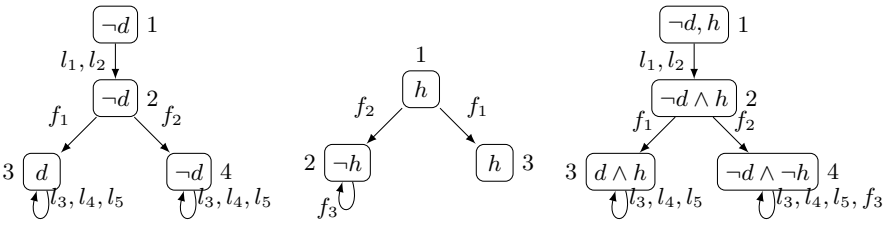


Fig. 3. Left and middle: the semantics of modules from Example 1 reduced w.r.t. $\mu X.(\langle A\rangle X \vee (h \wedge d))$. For clarity, only labels involving d and h have been displayed, a label such as $\neg x$ denotes a real label where x is not involved and thus implies $\neg x$. Right: the synchronised product of the two LTS on the left.

We begin by reducing the first LTS w.r.t. the formula. We are in case 6 of definition 7, $Pass^{\mu X.(\langle A\rangle X \vee (h \wedge d))} = \emptyset$ (we can never conclude without knowing the value of h which is and external variable) and $Fail^{\mu X.(\langle A\rangle X \vee (h \wedge d))} = \{10, 11, 12\}$ (we know that from these states we can only access states where d is false). We now apply function $\mathfrak{F}^{\langle A\rangle X \vee (h \wedge d)}$ repeatedly starting from:

- $(p_0, f_0, r_0) \stackrel{\text{df}}{=} (\emptyset, \{10, 11, 12\}, \{\{10, 11, 12\}, \{1, 2, 3, 4, 5, 6, 7, 8, 9\}\})$ where r_0 is given as the set of its equivalence classes instead of as a set of pairs, which is more compact;
- $\mathfrak{F}^{\langle A\rangle X}(p_0, f_0, r_0) = (\emptyset, \{10, 11, 12\}, \{\{10, 11, 12\}, \{1\}, \{2, 3\}, \{4, 5, 6, 7, 8, 9\}\})$ and $\mathfrak{F}^{h \wedge d} \stackrel{\text{df}}{=} (\emptyset, \{10, 11, 12\}, \{\{10, 11, 12\}, \{1, 2, 3, 4, 5, 6, 7, 8, 9\}\})$ so we have $(p_1, f_1, r_1) \stackrel{\text{df}}{=} \mathfrak{F}^{\langle A\rangle X \vee (h \wedge d)}(p_0, f_0, r_0) = (\emptyset, \{10, 11, 12\}, \{\{10, 11, 12\}, \{1\}, \{2, 3\}, \{4, 5, 6, 7, 8, 9\}\})$
- $(p_2, f_2, r_2) \stackrel{\text{df}}{=} \mathfrak{F}^{\langle A\rangle X \vee (h \wedge d)}(p_1, f_1, r_1) = (p_1, f_1, r_1)$ (We have reached the fixed-point so we can use r_1 to build the reduced LTS from figure 3)

Doing the same with the second LTS, we get the reduced LTS depicted in figure 3. Their product is depicted on the right of the same figure. To compose this product with the rest of the system, we shall first try to minimise it. Doing so, we also compute $\mathsf{Pass}^{\mu X.\langle A\rangle X \vee (h \wedge d)}$, obtaining set $\{1, 2, 3\}$ that contains the initial state of the graph. Therefore we know that the formula is true over the global system and we can stop the analysis. ◇

5 Conclusion

We have shown that it is possible to define the semantics of a modular Petri net as a hierarchical composition of the semantics of its modules taken in any order. At each step, a subset of modules is considered, and its semantics can be computed and analysed with respect to a modal μ-calculus formula φ. Possibly, this allows to draw a conclusion about the truth value of φ on the whole system without the need to consider the rest of the system. If no conclusion can be

drawn at this step, a minimised semantics can be computed for the subset of modules at hand, and reused for the sequel of the hierarchical analysis.

In [4], the authors define the decomposition of a Petri net according to a formula and the verification of this formula in a compositional way. Moreover, [3] makes use of the modular description of a system to reduce it hierarchically. The main difference with our work is that they both consider abstractions that preserve every formula from LTL\X in which chosen actions appear. Our approach only preserves one formula from the $\mu - calculus$ so, on the one hand, we can express more properties, and on the other hand, targeting a particular formula let us expects better reductions. But, as a consequence, we have to recompute the abstraction for each new formula. However more thoroughgoing comparisons remain to be done. In [7], the author considers the incremental construction of Petri nets through refinements (of transitions, places and place types), also allowing for incremental state space construction. Properties may be verified at an intermediary step avoiding to construct the fully refined state space. However, these properties are not expressed as logic formulae but are classical Petri net properties (deadlock, home state, etc.).

Future work will address the question of finding a good order for conducting such a hierarchical analysis, in order to minimise the computational effort needed to obtain a result. In particular, it is not clear if we should start by combining strongly connected modules with the aim of obtaining good reductions at the beginning, or if we should instead prefer the modules the most involved in the formula of interest. Another prospect is to find the best form for the formula we want to verify, in order too increase the efficiency of the reduction. Since we require equivalent states to be equivalent on every sub-formula, if we can minimize the number of sub-formulae then we are likely to compute a better reduction. For instance formula $\langle a \rangle true \vee \langle b \rangle true$ is equivalent to $\langle a, b \rangle true$. However if one state only has one outgoing transition labelled by a, and another state only has one outgoing transition labelled by b, they will be equivalent w.r.t. the second formula but not w.r.t. the first one.

So we believe that the current paper defines a framework that is suitable to perform hierarchical analysis, but also that it is the starting point of a lot more work to find suitable strategies for efficient analysis.

References

1. A. Aziz, T. Shiple, V. Singhal, R. Brayton, and A. Sangiovanni-Vincentelli. Formula-dependent equivalence for compositional ctl model checking. *Formal Methods in System Design*, 21(2):193–224, 2002.
2. S. Christensen and L. Petrucci. Modular analysis of Petri nets. *The Computer Journal*, 43(3):224–242, 2000.
3. K. Klai and L. Petrucci. Modular construction of the symbolic observation graph. In *Application of Concurrency to System Design, 2008. ACSD 2008. 8th International Conference on*, pages 88–97. IEEE, 2008.
4. K. Klai, L. Petrucci, and M. Reniers. An incremental and modular technique for checking ltl\ x properties of petri nets. *Formal Techniques for Networked and Distributed Systems–FORTE 2007*, pages 280–295, 2007.

5. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.
6. C. Lakos and L. Petrucci. Modular analysis of systems composed of semiautonomous subsystems. In *proc. of ACSD'04*. IEEE Computer Society Press, 2004.
7. G. Lewis. *Incremental specification and analysis in the context of coloured Petri nets*. PhD thesis, University of Tasmania, 2002.
8. A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.

A Proof of theorem 2

Theorem 2 can be rewritten using the property $\mathcal{P}$ defined, for any $(p, f, r) \in 2^Q \times 2^Q \times 2^{Q\times Q}$, by $\mathcal{P}(p, f, r)$ holds iff $(r \setminus (Q \setminus p)^2 \subseteq p^2$ and $r \setminus (Q \setminus f)^2 \subseteq f^2)$.

We want to prove that for any formula φ, we have $\mathcal{P}(\mathfrak{F}^\varphi)$. Let us show that this property is true for every base case and that it is preserved by the rules used for building $\mathfrak{F}^\varphi$.

Case $\mathfrak{F}^{\Sigma,B}$. Take (x, y) in $\sim^B$. We know that $L(x) \Rightarrow B \Leftrightarrow L(y) \Rightarrow B$. Then,

- if x (wlog) belongs to $\mathsf{Pass}^{\Sigma^p,B}$ then $L(x) \Rightarrow B$ is true, and so is $L(y) \Rightarrow B$ which means that y is in $\mathsf{Pass}^{\Sigma^p,B}$.
- if x belongs to $\mathsf{Fail}^{\Sigma^f,B}$ then $L(x) \Rightarrow B$ is false, and so is $L(y) \Rightarrow B$ which means that y is in $\mathsf{Fail}^{\Sigma^f,B}$.

Case $\mathfrak{F}^{\Sigma,\neg}$. Take (p, f, r) such that $\mathcal{P}(f, p, r)$. We then have $\mathcal{P}(\mathfrak{F}^{\Sigma,\neg}(p, f, r))$ because $\mathfrak{F}^{\Sigma,\neg}(p, f, r) = (f, p, r)$.

Case $\mathfrak{F}^{\Sigma,\vee}$. Take (p_1, f_1, r_1) and (p_2, f_2, r_2) such that $\mathcal{P}(p_1, f_1, r_1)$ and $\mathcal{P}(p_2, f_2, r_2)$. We have $\mathfrak{F}^{\Sigma,\vee}((p_1, f_1, r_1), (p_2, f_2, r_2)) = (p_1 \cup p_2, f_1 \cap f_2, r_1 \cap r_2)$.

- for the first case we have $(r_1 \cap r_2) \setminus (Q \setminus (p_1 \cup p_2))^2 = (r_1 \cap r_2) \setminus ((Q \setminus p_1)^2 \cap (Q \setminus p_2)^2) = (r_1 \cap r_2) \setminus (Q \setminus p_1)^2 \cup (r_1 \cap r_2) \setminus (Q \setminus p_2)^2 \subseteq r_1 \setminus (Q \setminus (p_1))^2 \cup r_2 \setminus (Q \setminus (p_2))^2 \subseteq p_1^2 \cup p_2^2 \subseteq (p_1 \cup p_2)^2$
- and for the second case $(r_1 \cap r_2) \setminus (Q \setminus (f_1 \cap f_2))^2 = (r_1 \cap r_2) \setminus (Q \setminus f_1 \cup Q \setminus f_2)^2 \subseteq (r_1 \cap r_2) \setminus ((Q \setminus f_1)^2 \cup (Q \setminus f_2)^2) \subseteq (r_1 \cap r_2) \setminus (Q \setminus f_1)^2 \cap (r_1 \cap r_2) \setminus (Q \setminus f_2)^2 \subseteq r_1 \setminus (Q \setminus f_1)^2 \cap r_2 \setminus (Q \setminus f_2)^2 \subseteq f_1^2 \cap f_2^2 \subseteq (f_1 \cap f_2)^2$

Case $\mathfrak{F}^{\Sigma,\langle\alpha\rangle}$. Take (p, f, r) verifying $\mathcal{P}$ and note $(p', f', r') \stackrel{\text{df}}{=} \mathfrak{F}^{\Sigma,\langle\alpha\rangle}(p, f, r)$. Then take $(x, y) \in r'$.

- If $x \in \mathsf{Pass}^{\Sigma^p,\langle\alpha\rangle}(p)$.
  Exists $x' \in p$ and $a \in \alpha \cap A^{loc}$ such that $x \xrightarrow{a} x'$.
  Because (x, y) belongs to r', we have $y \xrightarrow{a'} y'$ and $(x', y') \in r$. Since we know that $\mathcal{P}(p, f, r)$, y' belongs to p too and y to $\mathsf{Pass}^{\Sigma^p,\langle A\rangle}(p)$.
- If $x \in \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle}(f)$.
  - If x is in f then so is y because $\mathcal{P}(p, f, r)$ and $r' \subseteq r$. This is needed when $\alpha \cap Ex \neq \emptyset$.

- For all $y \xrightarrow{a} y'$ with a in α we have $x \xrightarrow{a'} x'$ with a' in α and (x', y') in r. Because $x \in \mathsf{Fail}^{\Sigma^f,\langle\alpha\rangle}(f)$ this means that every x' is in f. Then so is every y', and finally $y \in \mathsf{Fail}^{\Sigma^f,\langle A\rangle}(f)$.

Case $\mathfrak{F}^{\Sigma,X}$. We only put in the environment values verifying $\mathcal{P}$ (see next case).

Case $\mathfrak{F}^{\Sigma,\mu X.\varphi_1}$.

- $(\mathsf{Pass}^{\Sigma,\mu X.\varphi_1}, \mathsf{Fail}^{\Sigma,\mu X.\varphi_1}, r_0)$ with $(x, y) \in r_0$ iff
  1. $x \in Pass^{\Sigma^p,\mu X.\varphi_1}$ and $y \in Pass^{\Sigma^p,\mu X.\varphi_1}$ or
  2. $x \in Fail^{\Sigma^f,\mu X.\varphi_1}$ and $y \in Fail^{\Sigma^f,\mu X.\varphi_1}$ or
  3. both x and y belong to $Q \setminus (Fail^{\Sigma^f,\mu X.\varphi_1} \cup Pass^{\Sigma^p,\mu X.\varphi_1})$

  verify $\mathcal{P}$.
- $\mathfrak{F}^{\Sigma,\varphi_1}$ is a composition of the above functions, so it preserves proposition $\mathcal{P}$.

B Proof of theorem 3

Let $S \stackrel{\text{df}}{=} \bigotimes_{i\in I} S_i$ be a LTS , φ a formula and Σ_i environments we denote by π^φ the product relation of the $\sim_{S_i}^{\Sigma_i,\varphi}$, *i.e.*, (x, y) is in π^φ iff (x_i, y_i) is in $\sim_{S_i}^{\Sigma_i,\varphi}$ for all $i \in I$. Let us define the property $\mathcal{P}_\pi((p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \psi)$ which means: $p = \mathsf{Pass}^{\Sigma^p,\psi}$, $f = \mathsf{Fail}^{\Sigma^f,\psi}$ and $\pi^\psi \subseteq r$. We show that every base case verify this property and that the various rules preserve it. The part of the property about Pass and Fail can almost be obtained by construction so we do not explicitly mention it in this proof.

Case $\mathfrak{F}^{\Sigma,B}$. Take (x, y) in π^P. For all i in I we have the following property: $L_i(x_i) \Rightarrow B \equiv L_i(y_i) \Rightarrow B$. Now, we know that $\bigwedge_I L_i(x_i) \Rightarrow B \equiv \bigwedge_I L_i(y_i) \Rightarrow B$ and therefore $(x, y) \in \sim^{\Sigma,B}$. So we have $\mathcal{P}_\pi(\mathfrak{F}^{\Sigma,B}, \Sigma, \{\Sigma_i | i \in I\}, B)$.

Case $\mathfrak{F}^{\Sigma,\neg}$. Take (p, f, r) such that $\mathcal{P}_\pi((p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \varphi_1)$. We have $\mathcal{P}_\pi(\mathfrak{F}^\neg(p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \neg\varphi_1)$ because $\mathfrak{F}^\neg(p, f, r) = (f, p, r)$ and $(\mathsf{Pass}^{\neg\Psi}, \mathsf{Fail}^{\neg\Psi}, \pi^{\neg\Psi}) = (\mathsf{Fail}^\Psi, \mathsf{Pass}^\Psi, \pi^\Psi)$

Case $\mathfrak{F}^{\Sigma,\vee}$. For any formulae φ_1 and φ_2, for any (p_1, f_1, r_1) and (p_2, f_2, r_2) such that $\mathcal{P}_\pi((p_1, f_1, r_1), \Sigma, \{\Sigma_i | i \in I\}, \varphi_1)$ and $\mathcal{P}_\pi((p_2, f_2, r_2), \Sigma, \{\Sigma_i | i \in I\}, \varphi_2)$, we have $\mathcal{P}_\pi(\mathfrak{F}^{\Sigma,\vee}((p_1, f_1, r_1), (p_2, f_2, r_2)), \Sigma, \{\Sigma_i | i \in I\}, \varphi_1 \vee \varphi_2)$ because $\mathfrak{F}^{\Sigma,\vee}((p_1, f_1, r_1), (p_2, f_2, r_2)) = (p_1 \cup p_2, f_1 \cap f_2, r_1 \cap r_2)$ and $(\mathsf{Pass}^{\varphi_1\vee\varphi_2}, \mathsf{Fail}^{\varphi_1\vee\varphi_2}, \pi^{\varphi_1\vee\varphi_2}) = (\mathsf{Pass}^{\varphi_1} \cup \mathsf{Pass}^{\varphi_2}, \mathsf{Fail}^{\varphi_1} \cap \mathsf{Fail}^{\varphi_2}, \pi^{\varphi_1} \cap \pi^{\varphi_2})$

Case $\mathfrak{F}^{\Sigma,\langle\alpha\rangle}$. Take (p, f, r) such that $\mathcal{P}_\pi((p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \varphi_1)$.

Let us consider (x, y) in $\pi^{\langle A\rangle\varphi_1}$, it is indeed true that $(x, y) \in \pi^{\varphi_1}$. We now have take into account three different possibilities.

- If exists i such that x_i and y_i are in $\mathsf{Pass}_i^{\Sigma_i^p,\langle A\rangle\varphi_1}$ then x and y are in $\mathsf{Pass}^{\Sigma^p,\langle A\rangle\varphi_1}$.
- The same goes for Fail.
- In the third case:

- Let us consider any $x \xrightarrow{a} x'$ such that $x' \notin \mathsf{Fail}^{\Sigma^f,\varphi_1}$ and exists $i \in I$ such that $a \in \alpha \cap A_i^{fus}$, and let's show that exists y' such that $y \xrightarrow{a} y'$ and $(x', y') \in \pi^{\varphi_1}$.
  For every i in I:
  * If a is in A_i^{fus}: We have $x[i] \xrightarrow{a} x'[i]$. Because $(x[i], y[i])$ is in $\sim_i^{\Sigma,\langle A\rangle\varphi_1}$ and $x'[i] \notin \mathsf{Fail}_i^{\Sigma_i^f,\varphi_1}$, there exists y_i' such that $y[i] \xrightarrow{a} y_i'$ and $(x'[i], y_i') \in \sim_i^{\Sigma_i,\varphi_1}$
  * If a is not in A_i^{fus}: We have $x'[i] = x[i]$. Let's define $y_i' \stackrel{\text{df}}{=} y[i]$; we then have $(x'[i], y_i') \in \sim_i^{\Sigma_i,\varphi_1}$ because $(x'[i], y_i') \in \sim_i^{\Sigma_i,\langle A\rangle\varphi_1}$

  Now if $y'[i] \stackrel{\text{df}}{=} y_i'$ for all i, then $y \xrightarrow{a} y'$ and $(x', y') \in \pi^{\varphi_1}$.
- Let us consider any $x \xrightarrow{a} x'$ such that $x' \notin \mathsf{Fail}^{\Sigma^f,\varphi_1}$ and exists $i \in I$ such that $a \in \alpha \cap A_i^{loc}$. Let's show that exists $a' \in \alpha \cap A^{in}$ and y' such that $y \xrightarrow{a'} y'$ and $(x', y') \in \pi^{\varphi_1}$.
  * We know that we have $x[i] \xrightarrow{a} x'[i]$, and $a' \in (A_i^{loc} \cap \alpha)$ such that $y[i] \xrightarrow{a'} y_i'$ and $(x'[i], y_i') \in \sim_i^{\Sigma,\varphi_1}$
  * For $j \neq i$, we define $y_j' \stackrel{\text{df}}{=} y[j]$

  Now if $y'[i] \stackrel{\text{df}}{=} y_i'$ for all i, we have $y \xrightarrow{a'} y'$ with $a' \in (A^{loc} \cap \alpha)$ and $(x', y') \in \pi^{\varphi_1}$.

For any of these cases, (x, y) belongs to the third component of $\mathfrak{F}^{\Sigma,\langle\alpha\rangle}(p, f, r)$ so we have $\mathcal{P}_\pi(\mathfrak{F}^{\Sigma,\langle\alpha\rangle}(p, f, r), \Sigma, \{\Sigma_i | i \in I\}, \langle\alpha\rangle\varphi_1)$

Case $\mathfrak{F}^{\Sigma,X}$. We only put in the environment values which verify $\mathcal{P}_\pi(\Sigma, \{\Sigma_i | i \in I\}, X)$.

Case $\mathfrak{F}^{\Sigma,\mu X.\varphi_1}$.

– Let us show that if (x, y) is in $\pi^{\mu X.\varphi_1}$ and one of them is in $\mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}$ (resp $\mathsf{Fail}^{\Sigma^f,\mu X.\varphi_1}$) then so is the other. This means that we have the property $\mathcal{P}_\pi((p_0, f_0, r_0), \Sigma, \{\Sigma_i | i \in I\}, \mu X.\varphi_1)$, where (p_0, f_0, r_0) is the the starting point of the iteration. In order to do this we build the tuple (p, f, r) by iterating $\mathfrak{F}^{\Sigma,\varphi_1}$ starting from $(\emptyset, Q, Q^2)$. We then have $p = \mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}$, $f = \mathsf{Fail}^{\Sigma^f,\mu X.\varphi_1}$ and $r \supseteq \pi^{\mu X.\varphi_1}$ (This is the same proof we are currently doing, but with an easier base case). We can reuse the proof of theorem 2 (similarly, only the base case is different) to show that if $(x, y) \in r$ and one of them is in $\mathsf{Pass}^{\Sigma^p,\mu X.\varphi_1}$ (resp $\mathsf{Fail}^{\Sigma^f,\mu X.\varphi_1}$) then so is the other. So the property $\mathcal{P}_\pi(\Sigma^0(X), \Sigma^0, \{\Sigma_i^0 | i \in I\}, X)$ is true with:

  - $\Sigma^0 \stackrel{\text{df}}{=} \Sigma[X \leftarrow (p_0, f_0, r_0)]$ and
  - $\Sigma_i^0 \stackrel{\text{df}}{=} \Sigma_i[X \leftarrow (\mathsf{Pass}^{\Sigma_i,\mu X.\varphi_1}, \mathsf{Fail}^{\Sigma_i,\mu X.\varphi_1}, \sim_i^{\Sigma_i,\mu X.\varphi_1})]$ for all i.

– $\mathfrak{F}^{\varphi_1}$ is a composition of the previous functions. Therefore if the property $\mathcal{P}_\pi(\Sigma^n(X), \Sigma^n, \{\Sigma_i | i \in I\}, X)$ is true, then we have $\mathcal{P}_\pi(\mathfrak{F}^{\Sigma^n,\varphi_1}, \Sigma^n, \{\Sigma_i | i \in I\}, \varphi_1)$, which can be rewritten as $\mathcal{P}_\pi(\Sigma^{n+1}(X), \Sigma^{n+1}, \{\Sigma_i | i \in I\}, X)$ where $\Sigma^{n+1} = \Sigma^n[X \leftarrow \mathfrak{F}^{\Sigma^n,\varphi_1}]$. By recurrence, the property is true for any n.