# Algorithms and Hardness Results for Object Nets

Dissertation

zur Erlangung des akademischen Grades
Dr. rer. nat
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

eingereicht beim Fach-Promotionsausschuss Informatik von

Frank Andreas Heitmann

aus Hamburg (Deutschland)

Januar 2013

Gutachter:

Prof. Dr. Rüdiger Valk
Vertr.-Prof. PD Dr. Michael Köhler-Bußmeier
Prof. Dr.-Ing. Norbert Ritter
Univ.-Prof. Dr. Dr. h.c. Javier Esparza

Tag der Disputation: 3. Juli 2013

# Zusammenfassung

Im Zentrum dieser Arbeit steht die Frage, wie der Formalismus der elementaren Objektsysteme so eingeschränkt werden kann, dass die Möglichkeiten der Modellierung weitestgehend erhalten bleiben, Eigenschaften des Objektsystems aber automatisch und effizient überprüft werden können.

Elementare Objektsysteme sind Petrinetze, deren Marken eine innere Aktivität haben können und dazu selbst wieder als Petrinetz modelliert werden. Das entstehende Netzsystem hat dann eine zweistufige Struktur. Dieser von Valk eingeführte Formalismus und ähnliche Formalismen sind bei der Modellierung von Anwendungen nützlich, bei denen die Mobilität von Objekten und die Interaktionen zwischen diesen Objekten von Bedeutung sind.

Die Attraktivität eines solchen Formalismus wird durch die Möglichkeit, wichtige Verifikationsprobleme effizient lösen zu können, stark gesteigert. Da elementare Objektsysteme allerdings Turing-vollständig sind, muss der Formalismus eingeschränkt werden, wenn wichtige Probleme einer algorithmischen Lösung zugänglich sein sollen.

In dieser Arbeit werden zahlreiche strukturell und dynamisch eingeschränkte Varianten des Formalismus eingeführt und untersucht. Der Fokus liegt dabei auf der Komplexität des Erreichbarkeitsproblems, einem der wichtigsten Verifikationsprobleme. Durch die hohen unteren Schranken, die selbst für strukturell stark eingeschränkte Formalismen bewiesen werden können, wird deutlich, dass strukturelle Einschränkungen alleine nicht genügen, soll das Erreichbarkeitsproblem effizient gelöst werden. Für elementare Objektsysteme wird dann ein Sicherheitsbegriff eingeführt. Für sichere elementare Objektsysteme kann dann nicht nur das Erreichbarkeitsproblem, sondern jede Eigenschaft, die in den temporalen Logiken LTL oder CTL ausgedrückt werden kann, in polynomialen Platz entschieden werden. Dies gilt ohne weitere strukturelle Einschränkungen und tatsächlich haben diese nun kaum noch einen Effekt. Des weiteren kann für einen schwächeren Sicherheitsbegriff kein entsprechendes Resultat gezeigt werden.

Abweichend von der zentralen Fragestellung dieser Arbeit wird zuletzt der Formalismus der elementaren Objektsysteme erweitert. Objektnetzsysteme erlauben den vertikalen Transport von Marken, wodurch sich die Verschachtelungstiefe ändern kann. Des weiteren werden elementare Objektsysteme mit einer beliebigen, aber festen Verschachtelungstiefe eingeführt. Mit einem ähnlichen Sicherheitsbegriff wie zuvor kann für diese und für Objektnetzsysteme mit einem stärkeren Sicherheitsbegriff erneut bewiesen werden, dass das Problem der Modellprüfung für LTL und CTL in polynomialen Platz entschieden werden kann.

Sichere elementare Objektsysteme und stark sichere Objektnetzsysteme sind somit Formalismen, die es ermöglichen, Mobilität und Interaktionen zu modellieren, die es aber zugleich auch erlauben, wichtige Eigenschaften automatisch und mit einem vertretbaren Aufwand zu überprüfen.

# Abstract

The central question tackled in this thesis is, how the formalism of elementary object systems can be restricted, such that it retains most of its modelling features, but the verification of properties of the net system becomes feasible.

Elementary object systems are a Petri net formalism introduced by Valk. In elementary object systems the tokens of a usual Petri net are allowed to have an inner activity. To this end, these tokens are again modelled by Petri nets and thus a two levelled structure arises. This formalism and similar formalisms where Petri nets are somehow nested are helpful for the modelling of applications where the mobility of entities and the interaction between them are of importance.

The ability to solve important verification problems automatically and efficiently makes such a formalism far more attractive for practical purposes. However, being Turing-complete, elementary object systems need to be restricted if algorithmic solutions to important verification problems ought to be found.

In this thesis, several structural and dynamic restrictions of the formalism are introduced. The focus is then on the complexity of the reachability problem as one of the most prominent verification problems. Evident in the severe complexity bounds we establish even for structurally heavily restricted formalisms is that these restrictions alone are not enough to solve the reachability problem quickly. For elementary object systems with a safeness constrain we then show that not only reachability but every property expressible in the temporal logics LTL or CTL can be decided in polynomial space. This result holds without any further structural restrictions and we even show that such further restrictions have little to no effect. We also argue that other safeness constrains are too weak to establish a similar result.

Diverging from the central question, we also enhance the formalism of elementary object systems to object net systems. These allow the vertical transport of tokens and thus the nesting depth might change. We use these to introduce elementary object systems with an arbitrarily but fixed nesting depth and show that for these a similar safeness definition as for elementary object systems can be introduced. For these and for object net systems with a stronger safeness notion we establish a similar result as above, namely that LTL and CTL model checking is possible in polynomial space.

Safe elementary object systems and strongly safe object nets thus are formalisms, which allow to model mobility and interaction, but which also allow the automatic verification of properties, using an affordable amount of resources.

# Acknowledgement

This thesis was written while I was a member of the TGI group at the University of Hamburg. I am deeply grateful to Prof. Dr. Rüdiger Valk for the way he lead the group giving all members the possibility to pursue their scientific goals in an enjoyable and free atmosphere. His encouraging statements concerning my research have always helped me to continue. I am also deeply grateful to Prof. Dr. Matthias Jantzen and Prof. Dr. Manfred Kudlek. Matthias awoke my interest in Algorithms and especially in Complexity Theory while I was in my third semester and has accompanied my studies ever since. Manfred, who passed away unexpectedly shortly before the Petri Net conference in Hamburg in 2012, was always a cheerful supporter of my research and I am grateful to got to know him and his view on theory and theory research.

Above all I am grateful to Dr. Michael Köhler-Bußmeier who introduced me to the topic of this thesis right from the start in 2008 and has kept me on course through all the years, through the good times and - much more importantly - through the hard times. He was and is to me a colleague, mentor, and close friend.

I thank all the other members of the TGI group past and present for the many great discussions and conversations about research and also private life. Thank you Daniel, Lawrence, Gila, Thomas, Sofiane, Margit, Françoise, Berndt, Heiko, Jan, Michael, Tobias, Matthias, Marcin, Erik, Julia, Kolja, and Christine.

A great part of the writing of this thesis was done in the summer of 2012. During this time I met regularly with my colleague and friend Petra who wrote on her own thesis. Without her, writing would have much less fun and together we even got pleasantly through the times where you simply have no idea how to put the things you have in mind on the page. Thank you a lot, Petra!

I thank Prof. Dr.-Ing. Norbert Ritter who was so kind to write a review in addition to his tremendous work load. For the same reason I also thank Prof. Dr. Dr. h.c. Javier Esparza and I thank Prof. Dr. Ulrike von Luxburg for chairing my examination committee.

A couple of great people read parts of this thesis and send me their extremely helpful comments. I did not give them much time for this but they nonetheless did great work and spotted mistakes which I would have been ashamed of if they would have found their way in the final version. Moreover they send me encouraging comments helping me through the last, stressful days and nights. I thank Lorna, Erik, Georg, Christohper, Jan Henrik, Matthias, Sebastian, Thomas, Sascha, and my mother Ingrid for this hard work.

Special thanks go to Claudia. Claudia is not working in Computer Science. She is my Yoga teacher and friend and thanks to her I can not only twist my body in far more ways than before this thesis but my back came out unharmed of the writing process and the several hours I sat hunched in front of my laptop.

Last but not least, I would like to thank my family and my friends and all the people who have been part of my life in the last years. Without the constant support of my family and friends this work would not have turned out the way it did and indeed I might have stopped it one time or the other. Thank you Sebastian, Mieke, Annika, Petra, Robin, Bianca, Jan, Claudia, Silke, Daniel, Christopher H., Christopher S., Hüseyin, Georg, Patrick, David, Sascha, Christin, Malte, Benjamin, Tobias M., Sonja, Yvonne, Tayfun, Jan Henrik, Paula R., Jenny, Clawes, Niklas, Paula B., Felix W., Marc, Mehdi, Christine, Dorle, Felix K., and the members of the TGI group again. Thank you to all the students I met during my courses and lectures and especially to those who came to support me during the defence of this thesis. Thank you Mom, thank you Dad, thank you Ralf, Nina, Marlin, Alba, and Kaya. I wanted to be a scientist, since I was a small kid and, for me, this is a dream come true. I would not have done it without you. Thank you all! Thank you a lot!

# Contents

# 1 Introduction

Today, computer systems play a crucial role in our everyday lives. This is probably most evident in the Internet and its plenty useful applications, which for many of its users have become part of their daily lives by now. But computer systems are also important in many other areas and they often fulfil their respective tasks unnoticed by its users. In a mobile phone, for example, very sophisticated hard- and software comes into play to convert the voice into binary information and to establish a communication link between two persons. In a modern car, embedded systems control almost every aspect, from the windscreen wiper to the ejection of the airbags that offer safety in case of an accident. And at home most machines like refrigerators or washing machines also contain some kind of hardware and software.

Nowadays, many of these systems are *distributed*. The computation is not done by a single machine anymore that is manually given some input and then produces some output. Instead computation is done by logically and in many cases also physically distributed processes communicating over some kind of network. Moreover, systems no longer just compute an output. Many of todays systems are *reactive*. They run indefinitely, interact with their environment, and react to it or to user's needs.

A new aspect of computation, that becomes more and more important in todays systems and that is neither captured by distribution nor reactivity, is *mobility*. Mobility means that an entity is in some way able to move. For example, a mobile phone has to switch from one base station to another while the human, that uses the phone, moves. A software agent might "move" in a network or maybe just between different areas on one machine. A code snippet might, for example, traverses a firewall. Consequently, mobility and movement is not only meant physically.

It is to be suspected that mobile systems will be of crucial importance in the future. Computational devices, like cell phones, are carried around by people. They are embedded in other moving artefacts, like chips in a car, or are, by construction, moving entities by themselves, like robots. Moreover, they interact with humans or with other computational devices, making the systems even harder to understand. Suitable modelling languages and the ability to verify properties of the model are two key factors for designing such systems correctly.

To come up with such a modelling language, many formalisms have been proposed in the last decade, that try to capture the concept of mobility, an

aspect that was usually at least tedious to model using formalisms that were available at that time. Several of these formalisms apply in one way or another the idea of nesting to Petri nets, that is, they allow to interpret the tokens of an ordinary p/t net as p/t nets again. Certain formalisms of this kind are the subject of this thesis. Using an extension of Petri nets and thus a formal modelling language, one hopes that verification techniques can be developed and employed in a similarly successful fashion as was the case for Petri nets.

While Petri nets have been introduced some time ago in the seminal work by Carl Adam Petri [Pet62], formalisms of the kind mentioned above have only recently been introduced. Some examples are object nets [Val98], nested nets [Lom00], MOB nets [Kum00], adaptive workflow nets [LvHO+06], Petri hypernets [BBPP05], recursive nets [HP99], $PN^2$ [Hir02], Mobile Systems [Lak05], AHO systems [HEM05], and Hornets [KB09]. Another line of research, also dealing with nesting, but not in the field of Petri nets, is concerned with process calculi. Arguably most prominently among these calculi are the $\pi$-calculus by Milner [Mil99], the Ambient Calculus by Gordon and Cardelli [CG00b] and the Seal Calculus [CVN05] among many others. In Section 3.4 these formalisms are discussed and compared in more detail.

All these formalisms are helpful for modelling mobility of and interaction between different objects or agents. For example, at the latest *International Conference on Application and Theory of Petri Nets* Cristini and Tessier [CT12] presented work on modelling of innovative space architectures, for which they used *reference nets*, a formalism closely related to object nets and to the formalisms investigated in this thesis. Ma, Tsai, and Murata have formally modelled a secure mobile agent system, using an object net formalism that they extended according to needs in a security setting [MTM04, MT08]. Devarashetty, Tsai, Ma, and Zhang have used the same formalism to model a secure sensor network system [DTMZ08, DTMZ10].

However, despite the success in modelling a variety of applications, which are rather complicated to model with ordinary Petri nets, to say the least, verification issues have not been central to investigation yet. Moreover, many of the formalisms mentioned are Turing-complete and thus need to be carefully restricted before algorithmic solutions to verification question can be considered. The borderline, however, between modelling power on the one hand and complexity of the algorithms applied for verification issues on the other is by far not as well understood as for Petri nets. There free-choice Petri nets can be seen as the formalism that allows to model a high diversity of applications, while retaining a modest and manageable degree of complexity.

Since the systems, one attempts to model using the above mentioned formalisms, tend to be extremely complex, the ability to analyse the models and to verify automatically if specified properties are fulfilled by them or not, is of outmost importance. This is even more the case if expensive or safety-critical systems are subject to the modelling process.

This leads to the questions, how these formalisms can be restricted and how to solve important verification problems for these restricted formalism quickly – if at all.

In this thesis we tackle these questions for the aforementioned *Object Nets*, specifically for elementary object net systems with a distributed token semantic. Object nets are a Petri net formalisms introduced by Valk [Val91], [Val98] and are along with the $\pi$-calculus probably one of the most widely known formalisms for the modelling of mobile systems. In object nets a token may itself be a Petri net, allowing not only a nesting of structures, but also to capture mobility, thinking of the "net token" to be an agent (resp. a model thereof) that "moves" through the net in which it resides. So far object nets have been used as a modelling tool (for example in an agent context [KMR03]) and have been subject to decidability issues (cf. [Köh04]). But until now, the complexity of problems like reachability and liveness, that are of utmost importance in a verification context, have not been central to investigation. Furthermore, restrictions, like in the case of standard Petri nets, have not yet been thoroughly investigated neither.

This thesis aims at filling this gap by adopting known restrictions from Petri nets and also introducing new ones, only sensible for object nets. For these restricted formalisms we then study the complexity of the reachability problem as one of the most prominent verification problems with great significance for practical applications. We take little detours now and then and explore other significant problems, like the liveness problem, too.

The goal of this work is to attain a formalism that still captures the notion of mobility and that is still expressive in terms of modelling power, i.e. many applications or at least those, one is interested in, can still be modelled comfortably, while on the other hand important verification problems, like reachability and liveness, can be decided using affordable resources. The focus of this work is on the restriction of the formalism and the algorithmic solution of the reachability problem. The usage of the formalism, in terms of modelling itself, is not central to this work.

We start with *Elementary Object Net Systems* (EOS for short), which were already known before this thesis. EOS allow tokens to be Petri nets again. The uppermost net is the *system net*, other nets residing on its places are *net tokens* or *object nets*. The *nesting depth* of an EOS is restricted to two levels, that is no token of a net token is allowed to be a net token again, instead all tokens of a net token are black tokens. Despite this restriction of the nesting, the formalism is already Turing-complete. We slightly modify the construction from [Köh07] to prove this, but also give a new proof, which is reusable in the context of *conservative* EOS. Conservative EOS are restricted in such a way that every net token *type* in the preset of a transition also has to appear in its postset. We show that in this formalism, albeit not Turing-complete, reachability and liveness remain undecidable. We then turn to *Generalised*

*State Machines* (GSMs), which take the restriction utilised for conservative Eos one step further by demanding that for each transition $t$, each type either appears once in the preset *and* the postset of $t$ or neither appears in the preset nor the postset. Seemingly, a rather constrained specimen of the family of nested Petri net formalisms, GSMs allow to model situations in which the net tokens are agents, which are neither created nor destroyed and also not merged or divided. These assumptions are realistic for many physical or real-world systems.

A GSM can be "flattened" to a standard Petri net, the so called *reference net*, and so the standard problems are decidable. Unfortunately, the flattened Petri net can be huge even for small GSMs. We therefore introduce restricted GSMs to tackle this problem and aim *to attain a formalism that defines the border between expressiveness and analysability* in a similar way as free-choice Petri nets do for Petri nets.

While Eos, conservative Eos, and GSMs where known before (the proofs given here for the statements concerning conservative Eos are new, though) the restricted GSM formalisms are new. We introduce acyclic, conflict-free, and free-choice GSMs, which are GSMs, whose nets are restricted to be acyclic, conflict-free or free-choice, that is we *transfer these notions from general Petri nets to GSMs*, i.e. to an object net formalism. We also transfer the notion of P- and T-nets to GSMs.

Furthermore, we introduce *deterministic* and *strongly deterministic* Eos and GSMs. In these formalisms the possible communications resp. synchronisations between the system net and the object nets are restricted. These restrictions are only reasonable in the context of object net systems.

Taking combinations into account, these are already 30 *different structural restrictions*. For many of these we give upper and lower bounds regarding the complexity of the reachability problem.

All these structural restrictions usually have the advantage that fast algorithms exist to check, whether a given GSM has a certain structural property. Unfortunately, even many of the structurally severely restricted formalisms mentioned above tend to be of high complexity, that is, the reachability problem is not easily decidable. For example, restricting the GSM in such a way that the system net and all object nets are P-nets, i.e. each transition has exactly one place in its pre- and exactly one place in its postset, the reachability problem still requires polynomial space to be solved. In the case of free-choice GSM, the complexity even jumps to ExpSpace-hardness, even if the GSM is strongly deterministic and even if the reachability problem for the system net and all object nets in isolation could be solved in polynomial time. We therefore turn to dynamic restrictions of Eos. We take a look at unary and persistent Eos, two dynamical restrictions close to conflict-freedom, but both formalisms remain Turing-complete. Subsequently, we introduce the concept of *safeness* for elementary object net systems. We introduce four different vari-

ants of which two do not guarantee finiteness of the state space and indeed give rise to Turing-complete formalisms. The other two, however, do not only guarantee finiteness of the state space but also allow, by an adaption of a technique developed by Esparza for 1-safe Petri nets [Esp98a], to *decide every property expressible in the temporal logic CTL or LTL using only polynomial space* for this object net class. The requirement of safeness thus leads to a formalism of reasonable modelling power that can also be analysed using affordable resources and thus fulfils the aforementioned goal of this thesis. Formalisms that mix structural restrictions with the concept of safeness are also investigated, but only rarely this improves the bound below PSPACE.

Finally, we detour from the main track of this thesis and instead of restricting, *enhance* the formalism of EOS. We introduce EOS and GSMs with an arbitrarily, but fixed nesting depth and we introduce *Object Net Systems* (ONS), which allow a *vertical transport* of net tokens. In this way, the nesting of net tokens may change. From a modelling point of view, this is an interesting concept, because one can model, for example, that an object modelled by a p/t net is carried by an agent, also modelled by a p/t net, and that the object is then handed over to another agent, again modelled by a p/t net. Unsurprisingly, EOS with a fixed nesting depth and ONS are Turing-complete. However, for versions that have certain safeness restrictions, it is once again possible to show PSPACE-results similar to the ones mentioned above for safe EOS by use of the techniques established there. These *strongly safe ONS* then have additional modelling features compared to EOS as described above, lack some others, most notably the creation of net tokens, but therefore again allow the verification of properties within an acceptable complexity bound.

### Naming Convention

In the following, we will use the term *Petri net* or *general Petri net* for the unrestricted variant of a place/transition net, consisting of places, transitions, and weighted arcs and using only black tokens (see Definitions 2.24 and 2.27). We will also use the term *Petri net* if a structure can be *seen* as a Petri net if certain features are ignored. For example, a object net of an elementary object net system usually uses some channels to synchronise with the system net, but ignoring these channels (which are simply a labelling of the transitions) the object net can be seen as a Petri net. A similar reasoning applies to other nets like coloured Petri nets, the system net of a elementary object net system and so on. The meaning will always be clear from the context.

We will speak of the *nets-within-nets-family* of formalisms if we want to denote a formalism that somehow applies the idea of nesting to Petri nets (see the discussion of Petri net like formalisms in Section 3.4).

The term *object net system* is used to emphasise that we mean a member of the nets-within-nets-family, where the black tokens of a Petri net can be

a Petri net again or a reference to a Petri net. In particular, the formalisms introduced in this thesis are object net systems. The term is also used for the most general form we introduce in Chapter 6.

## Outline and Contribution

In the following we give an outline of the chapters of this thesis, relate the topics and results mentioned in the introduction above to the appropriate chapters, and list the publications that resulted from the work on this thesis and in which several results have already been published.

Figure 1.1 shows the chapter dependencies, in which the dashed lines indicate only minor dependencies.

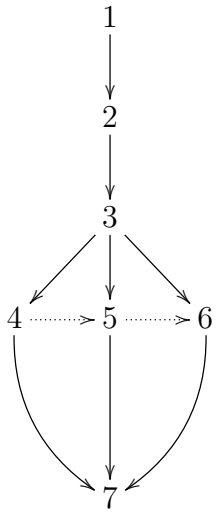


Figure 1.1: Chapter dependencies

After the introduction in this chapter, we review the most important notions and results for our purposes from complexity theory, logic and model checking, and from the theory of Petri nets in Chapter 2.

In Chapter 3 we motivate and introduce the formalisms of *Elementary Object Nets* and show that this object net formalism is Turing-complete in two different ways: A simulation of counter programs and a simulation of inhibitor nets. Also in Chapter 3 we discuss related approaches that also try to capture a nesting of structures and mobility of objects.

In Chapter 4 we investigate several structural restrictions of Eos and focus on the reachability problem for each of the introduced formalisms. We introduce conservative Eos, GSMs, deterministic and strongly deterministic GSMs, and GSMs that are obtained by restricting the participating nets to P-nets, T-nets, acyclic nets, conflict-free nets, and free-choice nets. By reduction we show that

the reachability problem becomes hard to decide, even for structurally very restricted formalisms.

In Chapter 5 we investigate dynamic restrictions of Eos. We introduce unary Eos, persistent Eos, and safe Eos. While the first two formalisms remain Turing-complete, we show in Section 5.3 that for safe Eos every property, expressible in the temporal logics CTL or LTL, can be decided in polynomial space. In Section 5.4 we discuss the effect of further structural restricting safe Eos.

In Chapter 6 we diverge from the path of the two previous chapters and, instead of restricting, *enhance* the formalism of Eos. We introduce Object Net Systems, a formalism in which the nesting depth is arbitrary and not bounded and in which it is possible to transport net tokens in the vertical dimension. It is shown that this formalism is Turing-complete, but that in a safe variant the reachability problem can again be solved in polynomial space. Furthermore, $\text{Eos}_k$ and $\text{GSM}_k$ are introduced, as a special case of object net systems, not allowing the vertical transport of tokens. $\text{Eos}_k$ and $\text{GSM}_k$ are then special Eos and GSMs with an arbitrary, but fixed, nesting depth $k$. We introduce a safeness notion for $\text{Eos}_k$ and show that for safe $\text{Eos}_k$ the techniques from Chapter 5 can be adapted and that again every property expressible in the temporal logics CTL or LTL can be decided in polynomial space, including the reachability problem.

Chapter 7 concludes this thesis, summarising the results obtained and gives an outlook to open questions and possible future work.

The subjects treated in Chapter 2 and most of Chapter 3 are not novel, but are included as the foundations upon which the results of this thesis are based. Furthermore, the formalism of conservative Eos and of generalised state machines defined in Sections 4.1 and 4.2 have also been introduced before this thesis. Some of the definitions mentioned above have been altered, though. For example, the definition of an Eos now contains the labelling instead of the set of events, which is more natural from a modelling point of view and also more suitable for complexity analysis. The results presented in Sections 3.3, 4.1, and 4.2 have also been known before, but several are proved here in a novel way.

All other chapters present my own work and work that was done jointly by me and Michael Köhler-Bußmeier. The construction of an Eos for a given inhibitor net presented in Section 3.3 has been published in [KBH12]. The work presented in Chapter 4 has been published in [HKB11a, HKB11b, HKB12b]. The work presented in Chapter 5 has been published in [KBH10b, KBH10a, KBH11b]. The work presented in Chapter 6 has been published in [HKB12a] and a similar but more complicated formalism, allowing the transport of net tokens in the vertical dimension, has been published earlier in [KBH09]. More details can be found in the respective chapters.

# 2 Fundamentals

In this chapter, we review the most important facts from algorithms, complexity theory, logic, model checking, and from the theory of Petri nets, which will be needed later on.

## 2.1 Notations and Fundamentals

Here we state some notational conventions, important definitions, and review some mathematical fundamentals. The presentation in this section is rather swift. We assume knowledge of fundamental mathematical objects introduced here, like sets and functions. Detailed explanations and a more rigorous treatment can be found in most introductory text books.

### Sets and Relations

The *natural numbers* are denoted $\mathbb{N}$. $\mathbb{N}$ does include 0. If we want to exclude 0 we explicitly write $\mathbb{N} \setminus \{0\}$. $\mathbb{Z}$ is the set of *integers*, $\mathbb{Q}$ the set of *rational numbers* and $\mathbb{R}$ the set of *real numbers*. With $\mathbb{Z}^+$, $\mathbb{Q}^+$, $\mathbb{R}^+$ the subsets with the numbers $x \geq 0$ are denoted, that is we have $\mathbb{Z}^+ := \{x \in \mathbb{Z} \mid x \geq 0\}$, for example. With $[n]$ we denote the set $\{1, 2, \ldots, n\}$, $\emptyset$ is the empty set.

Let $A$ and $B$ be sets. With $A \subseteq B$ we denote the *subset* relation. $A \subsetneq B$ is used to denote that $A$ is a *strict subset* of $B$. $A \cap B$ denotes the *intersection*, $A \cup B$ the *union* of sets. If an encompassing set $X$, i.e. a set $X \supseteq A$ is given, we denote with $\overline{A}$ the *complement* of $A$ with respect to $X$, i.e. the set $\overline{A} := \{x \in X \mid x \notin A\}$. With $\mathcal{P}(A)$ or $2^A$ we denote the *power set* of $A$, i.e. the set of all subsets of $A$, including $\emptyset$ and $A$. $|A|$ is the *cardinality* of $A$, i.e. the number of elements in $A$.

If $A_1, A_2, \ldots, A_n$ are sets, $A_1 \times A_2 \times \ldots \times A_n$ is the set of all *n-tuples* over $A_1, \ldots, A_n$. In $A_1 \times \ldots \times A_n$ are thus tuples $(a_1, \ldots, a_n)$ with $a_i \in A_i$, $i \in [n]$. The element $a_i$ in such a tuple is called the *ith component* of $(a_1, a_2, \ldots, a_n)$. If $A_1 = A_2 = \ldots = A_n = A$ we also write $A^n$ for the set $A_1 \times A_2 \times \ldots \times A_n$. A subset $R \subseteq A_1 \times A_2 \times \ldots \times A_n$ is called a *relation*. The tuple $(a_1, a_2, \ldots, a_n)$ is also called a *vector*, in which case the $A_i$ are usually the same set $A$.

For binary relations $R \subseteq A \times A$ we denote with $R^+$ the transitive and with $R^*$ the reflexive and transitive closure. Furthermore, we set $R^{-1} := \{(x, y) \mid (y, x) \in R\}$.

### Multisets

Let $A$ be a set. A *multiset over* $A$ is a mapping $ms_A : A \to \mathbb{N}$ assigning to each element $a$ in $A$ a number $ms_A(a) \in \mathbb{N}$ denoting how often $a$ appears in the multiset $ms_A$. The subscript $A$ can be omitted. Multisets are often written as sums: $\sum_{a \in A} ms_A(a) \cdot a$. A multiset $ms_A$ is *finite* if only finitely many members of $A$ are assigned a number greater than 0, i.e. if $\sum_{a \in A} ms_A(a) < \infty$.

Multisets over the same set can be added, subtracted and so on in the usual way for functions, e.g. $(ms_A + ms'_A)(a) = ms_A(a) + ms'_A(a)$ for all $a \in A$ is the addition of two multisets. Note that in the case of subtraction no value below 0 is possible, i.e. $(ms_A - ms'_A)(a) = \max\{0, ms_A(a) - ms'_A(a)\}$.

A multiset $ms_A$ is a *submultiset* of $ms'_A$, denoted $ms_A \leq ms'_A$, iff $ms_A(a) \leq ms'_A(a)$ for all $a \in A$.

The set of all multisets over a set $A$ is denoted by $\mathcal{MS}(A)$ or $\mathcal{MS}_A$.

### Vectors and Matrices

An *n-vector* is simply an $n$-tuple where all elements are from the same set $A$, i.e. each $x \in A^n$ is an $n$-vector. Vectors are written as columns $x = (a_1, \ldots, a_n)$, in which $a_i$ is the $i$th component of $x$ also denoted as $x_i$. To get a row vector we use the *transposed vector* $x^T$.

A collection of elements grouped together in $m$ rows and $n$ columns is an $m \times n$ *matrix* $A$. The element at the $i$th row and $j$th column is denoted by $A[i, j]$ or $A_{ij}$. If the elements are explicitly given as $A_{ij} := a_{ij}$, the matrix itself is then sometimes denoted as $A = [a_{ij}]$. If the $a_{ij}$ are from a set $B$ we write $A \in B^{m \times n}$ to denote that $A$ is an $m \times n$ matrix over the set $B$. The $n$-vector consisting of the elements of the $i$th row of $A$ is denoted by $A_i$ or $A[i, \bullet]$, i.e. $A_i = A[i, \bullet] = (a_{i1}, a_{i2}, \ldots, a_{in})$. The $m$-vector consisting of the elements of the $j$th column of $A$ is denoted by $A^j$ or $A[\bullet, j]$, i.e. $A^j = A[\bullet, j] = (a_{1j}, a_{2j}, \ldots, a_{mj})$.

The *transpose* of a matrix $A$ is denoted by $A^T$. Note that an $n$-vector is a special $n \times 1$ or $1 \times n$ matrix and that transposition converts one into the other. Vectors are written as $n \times 1$ matrices, so in matrix equations like $A \cdot x = b$ it is usually understood that $x$ and $b$ are $n$-vectors written as $n \times 1$ matrices, if $A$ is an $m \times n$ matrix. In case of $x^T \cdot A$, on the other hand, we have to use the transpose of $x$ and thus $x^T$ is an $n$-vector written as an $1 \times n$ matrix.

With $\mathbf{0}$ we denote the *zero vector* or *zero matrix*, depending on the context. With $I$ we denote the *unit square matrix*, i.e. the matrix defined by $I_{ij} := 1$ if $i = j$ and $I_{ij} := 0$ if $i \neq j$.

If $A = \{a_1, \ldots, a_n\}$ is a finite set we identify the multiset $ms : A \to \mathbb{N}$ with the vector $x = (ms(a_1), \ldots, ms(a_n)) \in \mathbb{N}^{|A|}$ and vice versa a vector $x \in \mathbb{N}^{|A|}$ with the multiset $ms : A \to \mathbb{N}$ where $ms(a_i) := x_i$.

### Language Theory

In language theory an *alphabet* $\Sigma$ is a finite set of *symbols*. The set $\Sigma^*$ consists of all finite *words* that can be formed by concatenating these symbols. With $\epsilon \in \Sigma^*$ the special word of length 0, the *empty word*, is denoted. The set $\Sigma^\omega$ additionally contains the words of *infinite length*. If $w \in \Sigma^*$ is a word, then the *length* of $w$ is denoted by $|w|$. Concatenation of two words $w, u \in \Sigma^*$ is denoted by $w \cdot u$ or simply $wu$. If $w \in \Sigma^*$ and $u \in \Sigma^\omega$ then $w \cdot u$ is possible, but not $u \cdot w$, if $u$ is of infinite length.

### Graphs

A *graph* is a tuple $G = (V, E)$ with a set $V$ of *vertices* or *nodes* and a set $E$ of *edges*. If $G$ is an *undirected* graph, $E$ consists of two-element subsets of $V$, i.e. $E \subseteq \{\{v, v'\} \mid v, v' \in V \text{ and } v \neq v'\}$. If $G$ is a *directed* graph, $E \subseteq V \times V$, i.e. $E$ consists of two-element tuples over $V$. A tuple $(v_1, v_2)$ means that an edge *starting* at $v_1$ and *ending* at $v_2$ exists. Vertices and edges can also be labelled. The former by a *labelling function* $L : V \to Lab$, where $Lab$ is a set of labels. The latter either by a similar labelling function or by adjusting the definition of edges slightly. A *directed labelled graph* for example is a tuple $G = (V, E, Lab)$ with $E \subseteq V \times Lab \times V$.

If $G = (V, E)$ is a directed graph, a sequence of nodes $v_0, v_1, \ldots, v_n$ such that $(v_i, v_{i+1}) \in E$ for all $i \in \{0, 1, \ldots, n-1\}$ is a *path* of length $n$. If no node appears more than once on a path, the path is called *simple*. If $v_0, v_1, \ldots, v_{n-1}$ is a simple path and an edge between $v_{n-1}$ and $v_0$ exists, then the sequence $v_0, v_1, \ldots, v_{n-1}, v_n$ is a *circle* or *cycle* of length $n$. The definitions for undirected graphs are analogous.

## 2.2 Algorithms and Complexity Theory

In this section, we review fundamental facts from computability and complexity theory and state the most important complexity classes needed later to relate the complexity of the reachability problems of the different net classes. More detailed explanations can be found in the text books [HMU01] and [Sip97] for fundamentals and in the text books [Pap94], [Rot05] for complexity theory in particular. Furthermore, the book by Hemaspaandra and Ogihara [HO02] gives a detailed account of many different complexity classes and reduction mechanisms.

One of the most fundamental and important formalism to reason about what is computable is the Turing machine, originally introduced by A. M. Turing in his seminal paper [Tur36] in 1936.

A Turing machine works on a one-way infinite tape consisting of cells. Each cell either holds a tape symbol or is empty, in which case it is marked with

the blank symbol. The Turing machine is in one of a finite set of states and has a tape head that is positioned above one cell. The tape head can read the symbol in the cell it is positioned above, write a new symbol to that cell and move to the left or to the right. At the beginning the input word $w$ is written on the tape, the Turing machine starts in its initial state and the tape head is positioned at the first symbol of $w$. The Turing machine then changes, depending on the current state and the tape symbol read, its state, writes a new symbol on the tape cell read and then moves the tape head to the left, to the right, or does not move it at all. The Turing machine then continues in this fashion. If during this computation the Turing machine reaches a final state, the input word is accepted.

**Definition 2.1** (**Turing machine**). *A* Turing machine *is a tuple*

$$M = (Q, \Sigma, \Gamma, \#, \delta, q_0, Q_{end}),$$

*where*

- $Q$ *is a finite set of states,*
- $\Sigma$ *is a finite alphabet of* input symbols*,*
- $\Gamma$ *is a finite alphabet of* tape symbols*, where* $\Gamma \supsetneq \Sigma$ *and* $\Gamma \cap Q = \emptyset$*,*
- $\# \in \Gamma \backslash \Sigma$ *is a special symbol describing the empty cell (the* blank symbol*),*
- $\delta$ *is a* transition function*,*
- $q_0 \in Q$ *is the* initial state*, and*
- $Q_{end} \subseteq Q$ *is the set of* finite states*.*

*In general* $\delta$ *is only a partial function. In case of a* deterministic *Turing machine the transition function is defined as* $\delta : (Q \times \Gamma) \to (\Gamma \times \{L, R, H\} \times Q)$ *where the meaning of* $\delta(q, a) = (a', X, q')$ *is that if* $M$ *is in state* $q$ *reading the symbol* $a$ *from the tape, it replaces* $a$ *with* $a'$*, moves the tape head to the direction given by* $X \in \{L, R, H\}$ *(move to the left, move to the right, or hold position) and changes to state* $q'$*.*[1]

*In case of a* nondeterministic *Turing machine the transition function is defined as* $\delta : (Q \times \Gamma) \to 2^{\Gamma \times \{L,R,H\} \times Q}$*, i.e. for each tuple from* $(Q \times \Gamma)$ *more than one action may be possible.*

*A* configuration *of a Turing machine is a tuple* $(u, q, v) \in \Gamma^* \times q \times \Gamma^+$*, meaning that the word* $uv$ *is currently written on the tape (where to the left of*

[1] Note that a movement to the left might not be possible if the tape head is already on the left end of the tape. We do not go into details here and refer the reader to [HMU01] instead.

*$u$ and to the right of $v$ only the blank symbol is written on the tape), the Turing machine is currently in the state $q$ and the tape head scans the first symbol of $v$. Configurations are also written as $uqv$.*

*A sequence of configurations where each configuration follows from the one before according to $\delta$, is called a* computation.

A Turing machine $M$ *accepts a word* $w$ if, given $w$ as input, $M$ reaches during its computation a configuration $(u, q, v)$ with $q \in Q_{end}$, i.e. if $M$ reaches a final state. The set of all accepted words is the *accepted language* $L(M)$ of $M$. The class of all languages accepted by Turing machines is denoted $\mathcal{RE}$, the class of *recursively enumerable* languages. The class of languages for which a Turing machine exists, that halts on *all* inputs, is denoted $\mathcal{REC}$, the class of *recursive* or *decidable* languages.

A Turing machine $M$ can also *compute a function* $f : \Sigma^* \to \Gamma^*$. In this case a word $w \in \Sigma^*$ is given as input, and $M$ halts in the configuration $qv$ where $q \in Q_{end}$ and $v = f(w)$ iff $f$ is defined on $w$. If a function $f$ can be computed in this way, $f$ is said to be *Turing-computable*. There is a close connection between word acceptance and function computing. If a Turing machine computes a function $f$ then it is also possible to construct a Turing machine that accepts the Language $L = \{(w, f(w))\}$ and conversely.

Many formalisms exists that are equivalent to Turing machines in computational power, i.e. if a language $L$ can be accepted by a Turing machine it can also be accepted by the other equivalent formalism. The equivalent formalism is also said to be *Turing-complete*. Two of these formalisms are *counter machines* and *counter programs*, which will be needed later to prove Turing-completeness of some object net formalisms.

**Definition 2.2** (**Counter machines**). *A* 2-counter machine *is a tuple*

$$M = (Q, \delta_{0,0}, \delta_{0,1}, \delta_{1,0}, \delta_{1,1}, q_0, Q_{end})$$

*where*

- $Q$ *is a finite set of* states,
- $q_0 \in Q$ *is the* initial state,
- $Q_{end} \subseteq Q$ *is the set of* final states,
- $\delta_{0,0} : Q \to (Q \times \{1\} \times \{1\})$ *is the transition function for the case that both counters are equal to zero,*
- $\delta_{0,1} : Q \to (Q \times \{1\} \times \{-1, 1\})$ *is the transition function for the case that the first counter is equal to zero and the second is greater than zero,*

- $\delta_{1,0} : Q \to (Q \times \{-1, 1\} \times \{1\})$ *is the transition function for the case that the first counter is greater than zero and the second counter is equal to zero, and*

- $\delta_{1,1} : Q \to (Q \times \{-1, 1\} \times \{-1, 1\})$ *is the transition function for the case that both counters are greater than zero.*

*A* configuration *is a triple* $(q, n_1, n_2) \in Q \times \mathbb{N} \times \mathbb{N}$ *representing the state* $q$ *the machine is currently in and the values* $n_1$ *and* $n_2$ *of the first and second counter, respectively. The successor configuration of* $(q, n_1, n_2)$ *is defined via* $\delta_{i,j}$ *where* $i = min(1, n_1)$ *and* $j = min(1, n_2)$*. With* $\delta_{i,j}(q) = (q', d_1, d_2)$ *the successor configuration of* $(q, n_1, n_2)$ *is then defined as* $(q', n_1 + d_1, n_2 + d_2)$*.*

The usage of $\delta_{i,j}$ as described in the definition above is actually a test for zero and the idea can be easily generalised to $k$-counter machines for a fixed $k \in \mathbb{N}$. Similar to 2-counter-automata *counter programs* can be defined.

**Definition 2.3** (**Counter programs**)**.** *A* $m$-counter program *consists of a finite, non-empty sequence of commands* $cmd_1; cmd_2; \ldots; cmd_n$ *and has access to* $m \geq 2$ *counters* $c_1, c_2, \ldots, c_m$*. There are four different commands:*

1. $c_j := c_j + 1, j \leq m$*, which increases the counter* $j$ *by* 1*,*

2. $c_j := c_j - 1, j \leq m$*, which decreases the counter* $j$ *by* 1*, if* $c_j > 0$*,*

3. `ifzero` $c_j$ `jump` $k_1$ `else jump` $k_2$*,* $j \leq m, k_1, k_2 \leq n$*, which tests the counter* $c_j$ *for zero and jumps to* $cmd_{k_1}$ *if so and to* $cmd_{k_2}$ *if not, and*

4. `halt`*, which terminates the program.*

*The last command is always the halt-command, i.e.* $cmd_n =$ `halt`*, and this is the only halt-command present in a* $m$*-counter program.*

*A configuration is a tuple* $C = (k, c_1, \ldots, c_m)$ *consisting of the current position* $k$ *in the program and the value of each counter. The initial configuration is given by* $C_0 = (1, 0, \ldots, 0)$*.*

*The successor configuration* $C' = (k', c'_1, \ldots, c'_m)$ *of* $C = (k, c_1, \ldots, c_m)$ *depends on the command* $cmd_k$*:*

1. *If* $cmd_k$ *is* $c_j := c_j + 1$*, then* $k' = k + 1$*,* $c'_j = c_j + 1$ *and* $c'_i = c_i$ *for all* $i \neq j$*.*

2. *If* $cmd_k$ *is* $c_j := c_j - 1$ *and* $c_j > 0$*, then* $k' = k + 1$*,* $c'_j = c_j - 1$ *and* $c'_i = c_i$ *for all* $i \neq j$*. If* $c_j = 0$ *the program halts with an error.*

3. *If* $cmd_k$ *is* `ifzero` $c_j$ `jump` $k_1$ `else jump` $k_2$*, then* $c'_i = c_i$ *for all* $i \leq m$ *and* $k' = k_1$ *if* $c_j = 0$ *and* $k' = k_2$ *otherwise.*

4. *If $cmd_k$ is* `halt`*, then the program terminates successfully. This is only possible if $k = n$.*

*The successor configuration is uniquely defined and the transition is denoted by $C \to C'$.*

In the light of the theory of computability the question if a language can be decided at all, is of interest. From a more practical point of view the question of the complexity of this decision procedure is of great importance. This is the field of complexity theory where the resources needed to decide a language are measured, most importantly time and space. We give the definitions with regard to Turing machines.

**Definition 2.4** (**Complexity Classes**)**.** *Let $M$ be a Turing machine halting on each input and (in the case of nondeterminism) on each computation. The time complexity of $M$ is a function $t : \mathbb{N} \to \mathbb{N}$, where $t(n)$ is the maximal number of steps needed on inputs of length $n$. We say $t(n)$ is the* running time *of $M$ or $M$* runs in time *$t(n)$. The space complexity is defined analogously, where $s : \mathbb{N} \to \mathbb{N}$ and $s(n)$ is the maximal number of tape cells visited by the read-/write-head on an input of length $n$.*

*The class* P *consists of all languages that can be decided by a deterministic Turing machine whose running time is bounded by a polynomial.*

*Similarly the class* NP *consists of all languages that can decided by a nondeterministic Turing machine whose running time is bounded by a polynomial.*

*The classes* PSpace *and* NPSpace *are defined analogously for deterministic and nondeterministic Turing machines with polynomial space bounds.*

While the question if P equals NP (widely believed not to be the case) is unresolved, the question if PSpace equals NPSpace is answered in the affirmative by Walter J. Savitch in [Sav70]:

**Theorem 2.5.**

$$\text{PSpace} = \text{NPSpace}$$

Another central concept is the concept of a *reduction*, establishing connections between different problems and thus allowing to derive decidability results from known ones and also to capture the notion of the "complexity" of a problem, most notably renowned in the theory of NP-completeness.

**Definition 2.6** (**Reduction**)**.** *Given two languages $A, B \subseteq \Sigma^*$ a* reduction *from $A$ to $B$, denoted by $A \leq B$ is a Turing-computable $f$ such that $x \in A \Leftrightarrow f(x) \in B$ holds, i.e. each input $x$ can be algorithmically transformed into a word $f(x)$ such that $x$ is an element of $A$ if and only if $f(x)$ is an element of $B$.*

With the definition of reduction known decidability or undecidability results for a given language can be carried over to new languages, if one can construct a suitable reduction.

**Lemma 2.7.** *1. Let $A, B \subseteq \Sigma^*$. If $A$ is undecidable and $A \leq B$, then $B$ is undecidable, too.*

*2. Let $A, B \subseteq \Sigma^*$. If $B$ is decidable and $A \leq B$, then $A$ is decidable, too.*

While in Definition 2.6 no restrictions where imposed on $f$ besides being Turing-computable, in complexity theory one is mostly interested in reductions that only use a certain amount of time or space. One of the most important class of reductions are those that are computable in polynomial time.

**Definition 2.8** (**Polynomial-time reduction**). *Let $A, B \subseteq \Sigma^*$. A* polynomial time reduction *from $A$ to $B$ is a reduction from $A$ to $B$ that can be computed in polynomial time. This is denoted by $A \leq^p B$.*

**Definition 2.9** (**NP-Hardness and NP-Completeness**). *A language $B$ is* NP-hard *if $A \leq^p B$ holds for all $A \in$ NP, i.e. each language $A$ in NP is polynomial-time reducible to $B$.*

*$B$ is* NP-complete *if $B$ is NP-hard and $B \in$ NP holds.*

The definition of PSPACE-*hardness* and -*completeness* is analogously to the definition of NP-hardness and -completeness.

The canonical NP- and PSPACE-complete problems are `SAT` and `QBF`, respectively, i.e. the problem of deciding if a given propositional formula is satisfiable or if a given quantified Boolean formula is satisfiable (cf. [HMU01]).

There is much more that could be said about Turing machines (or other Turing-complete formalisms), but for us it is enough to know that Turing machines are in some sense as powerful as common imperative programming languages like C, meaning that they can simulated each other with only a polynomial increase in running time. Indeed the *Church-Turing thesis* states that all reasonable deterministic computational models can pairwise simulate each other with only a polynomial overhead, meaning that if one of them runs in $t(n)$ time on inputs of length $n$, then another model can do the same computation in $O(p(t(n)))$ time, where $p$ is a polynomial. (We assume familiarity with the $O$-Notation. See for example [CLRS09].) Thus we will, as is common practice, not give Turing machines to prove decidability of a given problem, but describe an algorithm in an intuitively understandable pseudo code of an imperative-style programming language.

Since the algorithm can be simulated with only polynomial overhead by a Turing machine, a result that e.g. the algorithm requires only a polynomial number of steps and each step only requires polynomial time, also means that a Turing machine can calculate the algorithm in polynomial time, i.e. the problem is in P. A similar argumentation holds for the other classes employing

polynomials in their definitions, particularly for the classes NP, PSPACE, and NPSPACE.

## 2.3 Logic and Model Checking

In this section, we review basic notations and definitions from logic. Most notably we introduce the temporal logics LTL and CTL, which are widely used to specify a system's properties, and discuss the basics of model checking.

A more thorough account of propositional logic, linear time and branching time temporal logic, and model checking can be found in the wonderful text book by Huth and Ryan [HR04] and in the newer text book by Baier and Katoen [BK08].

### Propositional Logic

Propositional logic is defined over a countable set $V = \{x_1, x_2, \ldots\}$ of (propositional) *variables* or *atoms*. The *alphabet* of propositional logic, i.e. the set of symbols that may be used to construct formulas, then consists of the variables in $V$, the connectives $\wedge$, $\vee$, and $\neg$ (for "and", "or", and "not") and the parentheses ( and ). The syntax of propositional logic, i.e. the well-formed formulas is defined using this alphabet.

**Definition 2.10** (**Syntax of propositional logic**). *The (well-formed) formulas of propositional logic are defined inductively:*

1. *Each variable $x \in V$ is a formula.*
2. *If $\phi$ is a formula, then so is $\neg\phi$.*
3. *If $\phi$ and $\psi$ are formulas, then so are $(\phi \wedge \psi)$ and $(\phi \vee \psi)$.*
4. *Only formulas generated by finitely many applications of item 1 to 3 above are well formed formulas of propositional logic.*

*Alternatively the syntax of propositional logic can also be defined by the following grammar where $x$ represents a variable.*

$$\phi ::= x \mid \neg\phi \mid (\phi \wedge \phi) \mid (\phi \vee \phi)$$

*We omit parentheses in the usual way and use the common abbreviations $\phi \Rightarrow \psi := \neg\phi \vee \psi$, $\phi \Leftrightarrow \psi := (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$, $\top := (x \vee \neg x)$, and $\bot := \neg\top$ for* implication*,* biimplication*,* tautology*, and* contradiction*, respectively.*

*A formula only consisting of a single variable is also called an* atomic formula*. A formula of the Form $\neg\phi$ is called a* negation*. $(\phi \wedge \psi)$ is a* conjunction*, $(\phi \vee \psi)$ a* disjunction*.*

*A* literal *is an atomic formula or the negation of a atomic formula, i.e. a literal is of the form $x$ or $\neg x$ for an $x \in V$. In the first case the literal is also called* positive literal *and it is called a* negative literal *in the second case.*

*A formula is in* conjunctive normal form *(CNF) if it is a conjunction of disjunctions of literals. For example $(x_2 \vee \neg x_4) \wedge (x_1 \vee \neg x_4 \vee x_5) \wedge (\neg x_3 \vee x_4)$ is a* formula in CNF. *A disjunction of literals is also called a* clause. *A formula is in* 3-CNF *if each clause consists of exactly three literals.*

To define the semantics of propositional logic we need to define a model with regard to which the *truth* of a formula can then be defined.

**Definition 2.11** (**Propositional model**). *A* model *for a propositional formula $\phi$ is a total function $\mathcal{A} : V \to \{0, 1\}$.*

*If $\mathcal{A}(x) = 0$ we say, that $x$ is* false *(in $\mathcal{A}$). If $\mathcal{A}(x) = 1$ we say, that $x$ is* true *(in $\mathcal{A}$).*

Actually $\mathcal{A}$ does not need to be total, but only to be defined on the variables appearing in the formula $\phi$, but requiring $\mathcal{A}$ to be total eases some other definitions and the value of $\mathcal{A}$ on variables not appearing in $\phi$ is not important and might thus be defined arbitrarily. Thus these values are in general not mentioned explicitly.

The definition of $\mathcal{A}$ is expanded inductively to formulas. With $\mathcal{A} \models \phi$ we denote the truth of a propositional formula $\phi$ in a model $\mathcal{A}$, i.e. the fact that $\mathcal{A}(\phi) = 1$ holds. In case of $\mathcal{A}(\phi) = 0$ we write $\mathcal{A} \not\models \phi$.

**Definition 2.12** (**Semantic of propositional logic**). *Given a formula $\phi$ and a model $\mathcal{A}$, $\mathcal{A}$ is expanded to $\phi$ inductively by the following rules:*

$$
\begin{array}{lll}
\mathcal{A} \models p & \textit{iff} & \mathcal{A}(p) = 1, \textit{for a } p \in V \\
\mathcal{A} \models \neg\phi & \textit{iff} & \mathcal{A} \models \phi \textit{ does not hold, denoted as } \mathcal{A} \not\models \phi \\
\mathcal{A} \models \phi_1 \wedge \phi_2 & \textit{iff} & \mathcal{A} \models \phi_1 \text{ and } \mathcal{A} \models \phi_2 \textit{ holds} \\
\mathcal{A} \models \phi_1 \vee \phi_2 & \textit{iff} & \mathcal{A} \models \phi_1 \text{ or } \mathcal{A} \models \phi_2 \textit{ holds}
\end{array}
$$

*If $\mathcal{A} \models \phi$ holds, we say that $\mathcal{A}$* satisfies *$\phi$ or that $\phi$ is* true in *$\mathcal{A}$. Otherwise we write $\mathcal{A} \not\models \phi$ and say that $\mathcal{A}$* falsifies *$\phi$ or that $\phi$ is* false *in $\mathcal{A}$.*

*A formula $\phi$ is* satisfiable *if some model $\mathcal{A}$ exists with $\mathcal{A} \models \phi$. $\phi$ is falsifiable if some $\mathcal{A}$ with $\mathcal{A} \not\models \phi$ exists. $\phi$ is* valid *if $\phi$ is true in all models. Validity is denoted by $\models \phi$. In this case $\phi$ is a* tautology. *$\phi$ is a* contradiction, *denoted by $\phi \models$, if no model satisfies $\phi$.*

*Two formulas $\phi$ and $\psi$ are* equivalent, *if $\mathcal{A} \models \phi$ iff $\mathcal{A} \models \psi$ holds for all models $\mathcal{A}$.*

## The Temporal Logics LTL and CTL.

Temporal logics extend propositional logic by modalities like "finally" or "always" to reason about time. Other logics like predicate logic can also be extended in this way, but we focus on *Linear-time Temporal Logic* (LTL for

short) and *Computation Tree Logic* (CTL for short) here, which are extensions of propositional logic. The origins of LTL can be dated back to [Pnu81] (with an earlier conference version from 1977) where it was first used to reason about concurrent programs. The origins of CTL can be dated back to [CE81].

In LTL time is viewed as "linear", that is a state is followed by another, the *next* state. The infinite continuation is then a path. Each path might be the "actual" path that is realised. In CTL time might "branch", that is one state is followed by one of a number of possible successor states. Again there are different paths and any one of those might be the "actual" path that is realised. The logics are also enriched with quantifiers that range over paths.

In both logics it is not truly reasoned about time as in a statement like "the state $s$ is reached in 5 seconds", where a time duration is explicitly mentioned. Instead of the timing of events, the relative ordering of states is important and can be specified, for example one might state that a state is *eventually* followed by a state (in which a certain proposition holds).

In the following we will define the syntax and semantics of the two temporal logics LTL and CTL, which by now have both been successfully used in the verification of systems (see for example the book edited by Grumberg and Veith to celebrate 25 years of model checking [GV08]). The semantics are defined relative to a labelled transition system, which are basically directed graphs where each node, also called state in this setting, is labelled with atomic propositions that hold in that node.

We then state important results concerning model checking these logics. We focus on the logics CTL and LTL here. The logic CTL$^*$ that contains CTL and LTL as sublogics is not treated here (for an account of CTL$^*$ see for example [BK08]).

### The Temporal Logic LTL.

Let $V = \{v_1, v_2, \ldots\}$ be a countable set of *(propositional) variables* or *(propositional) atoms* as before. The syntax of LTL can be defined in two ways.

**Definition 2.13** (**Syntax of LTL**). *The (well-formed) formulas of LTL are defined by the following grammar*

$$\begin{aligned}\phi \quad ::= \quad & v \mid \neg\phi \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid \\ & X\phi \mid F\phi \mid G\phi \mid (\phi U \phi)\end{aligned}$$

*where $v \in V$ is any propositional atom.*

**Definition 2.14** (**Syntax of LTL (alternative)**). *The (well-formed) formulas of LTL are defined inductively:*

1. *Each atomic proposition $v \in V$ is an (atomic) formula of LTL.*

2. *If* $\phi_1$ *and* $\phi_2$ *are formulas, then so are* $\neg\phi_1, (\phi_1 \wedge \phi_2)$ *and* $(\phi_1 \vee \phi_2)$.

3. *If* $\phi_1$ *and* $\phi_2$ *are formulas, then so are* $X\phi_1, F\phi_1, G\phi_1$ *and* $(\phi_1 U \phi_2)$.

4. *Only formulas generated by finitely many applications of item* 1 *to* 3 *above are well formed formulas of LTL.*

To omit parentheses we use the convention that the unary connectives $\neg, X, G$ and $F$ bind most tightly, then $U$ and then $\wedge$ and $\vee$.

The logical connectives $\neg$, $\wedge$ and $\vee$ are as in propositional logic. The intended meaning of the new temporal connectives is "next", "globally", "finally", and "until" for $X$, $G$, $F$, and $U$ respectively. LTL formulas are interpreted along a *path* in a transition system (a formalism similar to a finite automaton; see below), where a path is simply a sequence of states. Before we define this formally we give the idea behind the new temporal connectives. Let $\pi$ be a path starting at state $s$. Intuitively $X\phi_1$ holds on $\pi$ if $\phi_1$ holds in the *next state* reached on $\pi$ after the first state $s$. $F\phi_1$ holds on $\pi$, if there is some state on $\pi$ in which $\phi_1$ holds. $G\phi_1$ holds on $\pi$, if $\phi_1$ holds in *all* states along $\pi$ (including $s$). At last $(\phi_1 U \phi_2)$ holds on $\pi$, if $\phi_2$ holds at a state $s'$ along $\pi$ and in all states until that state $\phi_1$ holds. More accurately the (sub-)formulas do not hold in states, but on paths starting at certain states. This is captured by the formal definition below. At first, however, we need the definition of a *transition system* along whose paths LTL formulas are interpreted.

**Definition 2.15** (**Labelled transition system (LTS)**). *A* labelled transition system *(LTS) is a tuple* $TS = (S, s_0, R, L)$ *where* $S$ *is a finite set of states,* $s_0 \in S$ *is a start state,* $R \subseteq S \times S$ *is a left-total transition relation, i.e. for each* $s \in S$ *there is always a* $s'$ *such that* $(s, s') \in R$*, and* $L : S \to \mathcal{P}(V)$ *is a labelling function, labelling each state* $s$ *with the set of atomic propositions* $L(s) \subseteq V$ *that hold in that state.*

Labelled transition systems as defined above are also called *Kripke structures* in honour of the philosopher and logician Saul Kripke who proposed them [Kri63]. The definition of a labelled transition system differs slightly in the literature. Frequently the arcs are in addition provided with an arc label, see e.g. [BK08].

**Definition 2.16** (**Path of an LTS**). *A* path $\pi$ *in an LTS* $TS = (S, s_0, R, L)$ *is an infinite sequence of states* $\pi = s_1 s_2 s_3 \ldots$ *such that* $(s_i, s_{i+1}) \in R$ *for all* $i \geq 1$*. With* $\pi^i$*,* $i \geq 1$*, we denote the suffix starting at* $s_i$*, i.e.* $\pi^i = s_i s_{i+1} \ldots$*, and with* $\pi(i)$*,* $i \geq 1$*, we denote the* $i$*-th state in* $\pi$*, i.e.* $\pi(i) = s_i$*. If* $s_1$ *happens to be the initial state* $s_0$ *of* $TS$*,* $\pi$ *is also called a* computation.

The semantics of LTL formulas is defined relative to paths in an LTS. The satisfaction relation $\models$ is again defined inductively.

**Definition 2.17** (**Semantics of LTL**). *Let $M = (S, s_0, R, L)$ be an LTS and $\pi = s_1 s_2 \ldots$ a path in $M$. $\pi$ satisfies an LTL formula $\phi$ (in $M$), if $M, \pi \models \phi$ holds, where the relation $\models$ is defined inductively as follows:*

$$
\begin{array}{lll}
M, \pi \models v & \textit{iff} & v \in L(s_1) \textit{ for } v \in V \\
M, \pi \models \neg\phi & \textit{iff} & M, \pi \not\models \phi \\
M, \pi \models \phi_1 \wedge \phi_2 & \textit{iff} & M, \pi \models \phi_1 \text{ and } M, \pi \models \phi_2 \\
M, \pi \models \phi_1 \vee \phi_2 & \textit{iff} & M, \pi \models \phi_1 \text{ or } M, \pi \models \phi_2 \\
M, \pi \models X\phi & \textit{iff} & M, \pi^2 \models \phi \\
M, \pi \models F\phi & \textit{iff} & M, \pi^i \models \phi \textit{ for some } i \geq 1 \\
M, \pi \models G\phi & \textit{iff} & M, \pi^i \models \phi \textit{ for all } i \geq 1 \\
M, \pi \models \phi_1 U \phi_2 & \textit{iff} & \textit{there is some } i \geq 1 \textit{ such that } M, \pi^i \models \phi_2 \textit{ holds} \\
 & & \textit{and for all } j < i \ M, \pi^j \models \phi_1 \textit{ holds.}
\end{array}
$$

A LTL formula $\phi$ holds in an LTS $M$, if it holds along *all* paths starting at $s_0$.

**Definition 2.18** (**Semantics of LTL (continued)**). *Let $M = (S, s_0, R, L)$ be an LTS. Let $\phi$ be a LTL formula and $s \in S$ a state of $M$. We write $M, s \models \phi$ if $M, \pi \models \phi$ holds for every path $\pi$ in $M$ that starts at $s$. If $M, s_0 \models \phi$ holds we also simply write $M \models \phi$ and say that $M$ is a* model *for $\phi$ or that $\phi$ is* satisfied *in $M$.*

*Two LTL formulas $\phi$ and $\psi$ are* equivalent*, denoted by $\phi \equiv \psi$, if for all models $M$ and all paths $\pi$ in $M$ we have $M, \pi \models \phi$ iff $M, \pi \models \psi$.*

Note that some authors introduce more connectives like $\Rightarrow$ for implication or the temporal connective $R$ for "release". These can be replaced by the equivalences $\phi_1 \Rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$ and $\phi_1 R \phi_2 \equiv \neg(\neg\phi_1 U \neg\phi_2)$ for example. The connectives we use are an adequate set of connectives for LTL, i.e. all other connectives can be expressed with them. Indeed there are smaller sets of adequate connectives, for example the set $\{\neg, \wedge, X, U\}$ is an adequate set. The connectives $F$ and $G$ are then defined via the abbreviations $F\phi := \top U \phi$ and $G\phi := \neg F \neg\phi$. Having only a small number of operators is helpful in devising (model checking) algorithms, since fewer cases have to be taken care of.

The model-checking problem for LTL asks given an LTS $M$ and a LTL formula $\phi$ if $M \models \phi$ holds, i.e. if $M$ is a model for $\phi$. It is possible to decide this problem with an automata-theoretic approach using Büchi automata. This approach yields an algorithm that solves the problem in $O(|M| \cdot 2^{|\phi|})$ time. The problem is actually PSPACE-complete. This is discussed in more detail in an extra section about model checking below. Before that section we introduce the temporal logic CTL.

### The Temporal Logic CTL.

In CTL it is possible to argue about the paths in a transition system. To this end the logic is enriched with the path quantifiers 'A' and 'E'. The semantics

is then defined with an infinite, directed tree in mind that is obtained by "unfolding" a transition system into a reachability tree (cf. [BK08]).

**Definition 2.19** (**Syntax of CTL**). *The (well-formed) formulas of CTL are defined by the following grammar*

$$\begin{aligned}\phi \quad ::= \quad & v \mid \neg\phi \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid \\ & EX\phi \mid EF\phi \mid EG\phi \mid E[\phi U \phi] \mid \\ & AX\phi \mid AF\phi \mid AG\phi \mid A[\phi U \phi]\end{aligned}$$

*where $v \in V$ is any propositional atom.*

*As in the case of LTL (see Definition 2.14) the formulas of CTL can also be defined inductively. We omit this here.*

The intended meaning of 'E' is "a path exists" the intended meaning of 'A' is "on all paths". The meaning of $EX$ is then "in some next state" and of $EF\phi$ "there is a path such that $\phi$ holds in some (future) state on it". Contrary to LTL formulas of CTL are evaluated with respect to *states* of an LTS. A CTL formula might thus hold in a state or not. Formally the semantics is again defined inductively.

**Definition 2.20** (**Semantics of CTL**). *Let $M = (S, s_0, R, L)$ be an LTS and $s \in S$ a state. A CTL formula $\phi$ is satisfied in $s$ (in $M$), if $M, s \models \phi$ holds, where the relation $\models$ is defined inductively as follows:*

| | | |
|---|---|---|
| $M, s \models v$ | *iff* | $v \in L(s)$ *for* $v \in V$ |
| $M, s \models \neg\phi$ | *iff* | $M, s \not\models \phi$ |
| $M, s \models \phi_1 \wedge \phi_2$ | *iff* | $M, s \models \phi_1$ and $M, s \models \phi_2$ |
| $M, s \models \phi_1 \vee \phi_2$ | *iff* | $M, s \models \phi_1$ or $M, s \models \phi_2$ |
| $M, s \models EX\phi$ | *iff* | *a state* $s' \in S$ *exists such that* $(s, s') \in R$ *and* $M, s' \models \phi$ *holds* |
| $M, s \models EF\phi$ | *iff* | *a path* $\pi = s_1 s_2 \ldots$ *starting at* $s$ *(i.e.* $s_1 = s$*) exists and an* $i \geq 1$ *such that* $M, s_i \models \phi$ *holds.* |
| $M, s \models EG\phi$ | *iff* | *a path* $\pi = s_1 s_2 \ldots$ *starting at* $s$ *(i.e.* $s_1 = s$*) exists and* $M, s_i \models \phi$ *holds for all* $i \geq 1$. |
| $M, s \models E[\phi_1 U \phi_2]$ | *iff* | *a path* $\pi = s_1 s_2 \ldots$ *starting at* $s$ *exists such that a* $j \geq 1$ *exists with* $M, s_j \models \phi_2$ *and* $M, s_i \models \phi_1$ *holds for all* $i < j$ |
| $M, s \models AX\phi$ | *iff* | $M, s' \models \phi$ *holds for all* $s' \in S$ *with* $(s, s') \in R$. |
| $M, s \models AF\phi$ | *iff* | *for all paths* $\pi = s_1 s_2 \ldots$ *starting at* $s$ *an* $i \geq 1$ *exists such that* $M, s_i \models \phi$ *holds.* |

| | | |
|---|---|---|
| $M, s \models AG\phi$ | *iff* | *for all paths* $\pi = s_1 s_2 \ldots$ *starting at* $s$ $M, s_i \models \phi$ *holds for all* $i \geq 1$. |
| $M, s \models A[\phi_1 U \phi_2]$ | *iff* | *for all paths* $\pi = s_1 s_2 \ldots$ *starting at* $s$ *a* $j \geq 1$ *exists such that* $M, s_j \models \phi_2$ *holds and* $M, s_i \models \phi_1$ *holds for all* $i < j$ |

A CTL formula holds in an LTS $M$, if it holds in the initial state of $M$.

**Definition 2.21** (**Semantics of CTL (continued)**). *Let* $M = (S, s_0, R, L)$ *be an LTS. Let* $\phi$ *be a CTL formula. If* $M, s_0 \models \phi$ *holds we also simply write* $M \models \phi$ *and say that* $M$ *is a* model *for* $\phi$ *or that* $\phi$ *is* satisfied *in* $M$.

*Two CTL formulas* $\phi$ *and* $\psi$ *are* equivalent*, denoted by* $\phi \equiv \psi$*, if for all models* $M$ *and all states* $s$ *in* $M$ *we have* $M, s \models \phi$ *iff* $M, s \models \psi$.

To exhibit the differences between LTL and CTL consider the following example taken from the book by Haubelt and Teich [HT10] (in German only). Consider the LTL formula $\phi = FGp$ (which is actually interpreted as $\phi = AFGp$), the CTL formula $\psi = AFAGp$ and the LTS $M$ given by

1. $S = \{s_0, s_1, s_2\}$
2. $R = \{(s_0, s_0), (s_0, s_1), (s_1, s_2), (s_2, s_2)\}$
3. $L(s_0) = L(s_2) = \{p\}$

Now, since LTL-formulas are interpreted over *individual paths* and CTL-formulas are interpreted in *states*, $\phi$ is true in the model, because it holds for every path (starting at $s_0$), but $\psi$ is not true in the model, because it would have to hold at $s_0$, i.e. at each path starting at $s_0$ one would have to reach a node in which $AGp$ holds, but this is not the case in $s_0$.

Thus even if the two formulas above seem alike, they are not. Indeed there are LTL-formulas for which no equivalent CTL-formula exist and also CTL-formulas for which no equivalent LTL-formula exists. Equivalence means here that the formulas have the same set of satisfying Kripke structures. Each of these logics is a sublogic of CTL$^*$, though, see e.g. [HR04] or [BK08].

We usually - and especially in the case of model checking - stick to a set of adequate connectives, i.e. a set of connectives that is expressive enough to capture the logic CTL. One such set is for example the set $\{\neg, \wedge, EX, EG, EU\}$. All formulas using any of the other connectives can always be replaced by equivalent formulas using only the connectives of the set $\{\neg, \wedge, EX, EG, EU\}$. Common abbreviations are $\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$, $\top \equiv \phi \vee \neg\phi$, $EF\phi \equiv E[\top U \phi]$, $AG\phi \equiv \neg EF\neg\phi$, $AF\phi \equiv \neg EG\neg\phi$, $AX\phi \equiv \neg EX\neg\phi$, and $A[\phi_1 U \phi_2] \equiv \neg(E[\neg\phi_2 U \neg(\phi_1 \vee \phi_2)] \vee EG\neg\phi_2)$. See [HR04] for more details.

Given a CTL formula $\phi$ and an LTS $M$, the model checking problem for CTL asks if $M \models \phi$ holds. The problem can be decided in $O(|M| \cdot |\phi|)$ time and is thus linear in the input. One should note, however, that the size of the

model $M$ tends to be rather big, giving rise to the infamous *state explosion* problem. We discuss CTL and LTL model checking in more detail in the following subsection.

## The Model Checking Problem

The *model checking problem* for LTL or CTL asks given an LTS $M$ and a formula $\phi$ if $M \models \phi$ holds, i.e. if $M$ is a model for $\phi$. CTL model checking was invented independently by E. M. Clarke and E. A. Emerson [CE81] and J.-P. Queille and J. Sifakis [QS82]. The ideas that underlie the method for LTL model checking outlined below were proposed by O. Lichtenstein and A. Pnueli [LP85] and M. Y. Vardi and P. Wolper [VW86].

Usually $M$ is a more or less abstract model of a system or of certain aspects of a system and $\phi$ is a specification of certain properties. If $M \models \phi$ holds, the (model of the) system satisfies the specification.

In order to state results concerning the complexity of decision procedures for the model checking problems, we need to define the size of an LTS and of a formula: Let $M$ be an LTS and $\phi$ be a CTL or LTL formula. The size of $M$ is denoted by $|M|$, the size of $\phi$ is denoted by $|\phi|$. We set $|M| := |S| + |R|$ and $|\phi| := n$, where $n$ is the number of atomic propositions in $\phi$. Note that each occurrence of an atomic proposition counts, e.g. the size of $\phi = (A \vee B) \wedge A$ is 3, since $A$ appears twice.

**Theorem 2.22** (**Model Checking LTL and CTL**). *Let $M$ be an LTS.*

1. *Let $\phi$ be a LTL formula. The model checking problem for LTL, i.e. the question if $M \models \phi$ holds, is* PSPACE*-complete and can be decided in $O(|M| \cdot 2^{|\phi|})$ time.*

2. *Let $\phi$ be a CTL formula. The model checking problem for CTL, i.e. the question if $M \models \phi$ holds, can be decided in $O(|M| \cdot |\phi|)$ time.*

The difference in terms of complexity in the procedures to decide the model checking problem for LTL and CTL stems from the fact that in CTL a formula is true in *a state* or not. Thus one can break down the given formula into its subformulas, test in which states these hold, and then use this to decide in which state the more complex, composed formulas hold. This is actually the main idea behind the basic CTL model checking algorithm (for a didactic treatment see [HR04]).

For LTL model checking, on the other hand, *all paths* have to be checked. Interpreting these paths as infinite words, a more language-theoretic approach seems worthwhile. Usually, a LTL model checker constructs a Büchi automaton $M_{\neg\phi}$ that accepts all words that do *not* fulfil the specification $\phi$. With the product construction for finite automata an automaton $M \otimes M_{\neg\phi}$ is then constructed that accepts all words that are in $L(M)$, i.e. that are possible runs in

the given LTS $M$, and that do not fulfil the specification. The model checking problem for LTL, that is the question if $M \models \phi$ holds, is thus reduced to the question whether the language of the automaton outlined above is empty, i.e. if $L(M \otimes M_{\neg\phi}) = \emptyset$ holds. The major task in this approach is the construction of $M_{\neg\phi}$. See [HR04] and [Var96] for details on this.

Note that regardless of the difference in complexity theoretic terms, the difficulty of solving the problems in practice usually stems from the size of the model and not the size of the formula. A model often consists of several components executed in parallel. The size of the model is then exponential in the number of such components. The size of the formula, on the other hand, tends to be rather small especially in comparison with the size of the model.

Despite the in most cases quite large model, both CTL and LTL model checking have been implemented in tools and have been successfully used in practice. For example the by now widely known model checker SPIN implements LTL model checking (see the SPIN book by Holzmann [Hol04]). CTL model checking is implemented in e.g. NuSMV [CCGR99]. The importance of model checking has grown since the early papers of E. M. Clarke and E. A. Emerson [CE81] and J.-P. Queille and J. Sifakis [QS82] in such a way, that for their research in model checking and their role in turning it into a verification technique widely used in the industry Clarke, Emerson and Sifakis received the prestigious ACM A.M. Turing Award in 2007.

## 2.4 Petri Nets

In this section, we state basic definitions and review some facts about Petri nets. Moreover, in this section we also treat extensions of Petri nets.

Several structurally and dynamically restricted variants are treated in Section 2.5. More restrictions and extensions can be found in the literature, but we focus on those variants that we will also treat later on in the context of object net systems.

More details on Petri nets can be found in e.g. [RR98], [GV03], or [Pet81].

### Basic Petri Nets

Petri nets are a by now well established formalism to model distributed systems and in particular their control flow. Originally introduced by Carl Adam Petri in his doctoral dissertation in 1962 [Pet62] they are now in widespread use. We will use the term *Petri net* or *general Petri net* for the unrestricted variant of a place/transition net We will also use the term, when a structure can be *seen* as a Petri net if certain features are ignored (see Definitions 2.24 and 2.27 below and the naming conventions in Chapter 1).

All Petri net models share the definition of a *net*:

**Definition 2.23 (Net).** *A* net *is a triple $N = (P, T, F)$, where*

1. *$P$ is a finite set of* places*,*
2. *$T$ is a finite set of* transitions *with $T \cap P = \emptyset$, and*
3. *$F \subseteq (P \times T) \cup (T \times P)$ is a* flow relation *describing the set of arcs*

*A net $(P, T, F)$ is* connected *if every two nodes $x, y$ satisfy $(x, y) \in (F \cup F^{-1})^*$.*

We do not distinguish between finite and infinite nets here. Our nets are are always finite, i.e. $|P|, |T| < \infty$.

With ${}^\bullet t$ we denote the set of places that have an arc to $t$, i.e. ${}^\bullet t = \{p \in P \mid (p, t) \in F\}$. Similarly we have $t^\bullet = \{p \in P \mid (t, p) \in F\}$. The sets ${}^\bullet p$ and $p^\bullet$ are defined analogously as the sets of transitions "before" and "after" a place $p$.

The class of place/transition nets (or p/t nets for short) is one of the most fundamental net classes and is defined as follows:

**Definition 2.24 (P/T net).** *A* place/transition net *(*p/t net *for short) is a tuple $N = (P, T, F, W)$, where*

1. *$(P, T, F)$ is a net, and*
2. *$W : F \to \mathbb{N} \setminus \{0\}$ is the* arc weight function*.*

If $W(x, y) = 1$ for all arcs $(x, y) \in F$ we usually omit $W$ in the tuple and simple write $(P, T, F)$ for a p/t net.

We will use the term *Petri net* or *general Petri net* for the place/transition net introduced above or if a structure can be seen as a place/transition net, if certain features are ignored.

An example of a p/t net is given in Figure 2.1.

In an alternative definition of p/t nets the arcs are given by the backward and forward incidence matrices **pre** and **post**. This notation is usually quite helpful when calculating invariants, whereas the usage of $F$ is closer to the graphical representation.

**Definition 2.25 (P/T net (alternative)).** *A* place/transition net *(*p/t net*) is a tuple $N = (P, T, \mathbf{pre}, \mathbf{post})$, where*

1. *$P$ is a finite set of* places*,*
2. *$T$ is a finite set of* transitions *with $T \cap P = \emptyset$, and*
3. *$\mathbf{pre}, \mathbf{post} \in \mathbb{N}^{|P| \times |T|}$ are the* backward *and* forward incidence matrices*.*

*The matrix $\Delta = \mathbf{post} - \mathbf{pre}$ is called the* incidence matrix *of $N$.*

*$\mathbf{pre}(t) := \mathbf{pre}[\bullet, t]$ and $\mathbf{post}(t) := \mathbf{post}[\bullet, t]$ are used as abbreviations.*

If there is an arc with weight $n > 0$ from $p \in P$ to $t \in T$, then we have $pre[p,t] = n$ or alternatively $(p,t) \in F$ and $W(p,t) = n$. Similarly for arcs from transitions to places. The two representations can thus easily be converted into each other. We usually use the second representation here.

**Definition 2.26 (Marking).** *A* marking *of a p/t net $N = (P, T, \mathbf{pre}, \mathbf{post})$ is a vector $m \in \mathbb{N}^{|P|}$ or alternatively a function $m : P \to \mathbb{N}$.*

*The set of all markings of a net $N$ is denoted by $\mathcal{M}_N$ or simply by $\mathcal{M}$ if no ambiguities can arise.*

*The special marking with $m(p) = 0$ for all $p \in P$ is denoted by $\mathbf{0}$, which is also used in the special case where $N$ has no places, i.e. $P = \emptyset$. In the last mentioned case there is thus only one marking and $\mathcal{M}_N = \{\mathbf{0}\}$.*

**Definition 2.27 (P/T net system).** *A* p/t net system *$\mathcal{N}$ is tuple $(N, m_0)$ consisting of a p/t net $N$ together with an* initial marking *$m_0$. Alternatively we write $\mathcal{N} = (P, T, \mathbf{pre}, \mathbf{post}, m_0)$.*

*If $\mathcal{N} = (P, T, \mathbf{pre}, \mathbf{post}, m_0)$ is a p/t net system, we denote the underlying p/t net $(P, T, \mathbf{pre}, \mathbf{post})$ by $N$.*

*In cases where no ambiguity can arise we use $\mathcal{N}$ and $N$ interchangeable. In the case of markings for example it does not matter if the set of markings is denoted by $\mathcal{M}_N$ or $\mathcal{M}_\mathcal{N}$.*

To state complexity results concerning problems where p/t net systems are the input the *size* of p/t nets systems needs to be defined.

**Definition 2.28 (Size of a p/t net system).** *Let $\mathcal{N} = (P, T, \mathbf{pre}, \mathbf{post}, m_0)$ be a p/t net system. The* size *of $\mathcal{N}$ is defined as $|\mathcal{N}| := |P| + |T| + |\mathbf{pre}| + |\mathbf{post}| + |m_0|$.*

Note that $|\mathbf{pre}|, |\mathbf{post}| \in O(|P| \times |T|)$ and also $|F| \in O(|P| \times |T|)$, so the different notions of a p/t net do not differ much in complexity terms. Also note that the size of $m_0$ depends on the number of markings that reside on each place. In an algorithm usually only **pre**, **post** and $m_0$ are given, where **pre** and **post** are two-dimensional arrays and $m_0$ is an one-dimensional array.

**Definition 2.29 (Enabled transition).** *Let $N = (P, T, \mathbf{pre}, \mathbf{post})$ be a p/t net. A transition $t \in T$ is* enabled *or* activated *in a marking $m \in \mathbb{N}^{|P|}$ if $m \geq \mathbf{pre}(t) = \mathbf{pre}[\bullet, t]$, where $\mathbf{pre}[\bullet, t] = (\mathbf{pre}[p_1, t], \mathbf{pre}[p_2, t], \ldots, \mathbf{pre}[p_n, t])$ for $P = \{p_1, \ldots, p_n\}$, i.e. $\mathbf{pre}[\bullet, t]$ is the $t$-column vector of the matrix* **pre**.

*If the net is given as $N = (P, T, F, W)$, $t$ is enabled in $m : P \to \mathbb{N}$ iff $m(p) \geq W(p,t)$ for all $p \in {}^\bullet t$.*

*Enabling of a transition $t$ in a marking $m$ is denoted by $m \xrightarrow{t}$.*

A transition $t$ that is enabled may *fire*. Firing removes the necessary amount of tokens from all places in the preset of $t$ and puts new tokens on all places in the postset of $t$ according to the arc weight function.

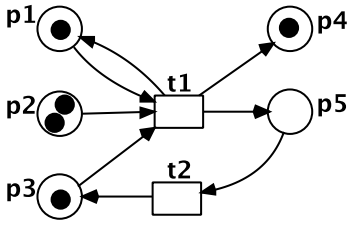


Figure 2.1: A p/t net

$$
\begin{aligned}
& p_1 + 2 \cdot p_2 + p_3 + p_4 \\
\xrightarrow{t_1} \quad & p_1 + p_2 + 2 \cdot p_4 + p_5 \\
\xrightarrow{t_2} \quad & p_1 + p_2 + p_3 + 2 \cdot p_4 \\
\xrightarrow{t_1} \quad & p_1 + 3 \cdot p_4 + p_5 \\
\xrightarrow{t_2} \quad & p_1 + p_3 + 3 \cdot p_4
\end{aligned}
$$

Figure 2.2: A firing sequence for the p/t net in Figure 2.1

**Definition 2.30** (**Firing a transition**). *A transition $t$ of a p/t net $N = (P, T, \mathbf{pre}, \mathbf{post})$ that is enabled in a marking $m \in \mathbb{N}^{|P|}$, i.e. $m \xrightarrow{t}$, may* fire. *The* successor marking *$m'$ is defined as $m' := m - \mathbf{pre}(t) + \mathbf{post}(t) = m - \mathbf{pre}[\bullet, t] + \mathbf{post}[\bullet, t]$, where $\mathbf{post}[\bullet, t]$ is defined analogous to $\mathbf{pre}[\bullet, t]$ (see Definition 2.29). We denote firing by $m \xrightarrow{t} m'$.*

*Again, if the net is given as $N = (P, T, F, W)$ the successor marking $m'$ of $m$ reached by firing $t$ is defined by*

$$
m'(p) = \begin{cases}
m(p) - W(p,t) & \text{if } p \in {}^\bullet t \setminus t^\bullet \\
m(p) + W(t,p) & \text{if } p \in t^\bullet \setminus {}^\bullet t \\
m(p) - W(p,t) + W(t,p) & \text{if } p \in {}^\bullet t \cap t^\bullet \\
m(p) & \text{otherwise}
\end{cases}
$$

To get rid of the case distinction we introduce a function $\tilde{W} : (P \times T) \cup (T \times P) \to \mathbb{N}$ with

$$
\tilde{W} = \begin{cases}
0 & \text{if } (x,y) \notin F \\
W(x,y) & \text{if } (x,y) \in F
\end{cases}
$$

The successor marking can now be easily defined as

$$
m'(p) = m(p) - \tilde{W}(p,t) + \tilde{W}(t,p)
$$

Firing is extended to sequences in the usual way:

**Definition 2.31** (**Firing a sequence of transitions**). *Let $w \in T^*$ a sequence of transitions. The firing of $w$ in a marking $m$ is defined inductively on the length of $w$:*

- $m \xrightarrow{\epsilon} m$
- *If $w = ut$ with $u \in T^*$ and $t \in T$, then $m \xrightarrow{w} m'$ iff a marking $m''$ exists such that $m \xrightarrow{u} m''$ and $m'' \xrightarrow{t} m'$ holds.*

*To denote, that $w \in T^*$ is activated in $m$ we write $m \xrightarrow{w}$. To denote that $m'$ is reachable from $m$ by some sequence of transitions we also write $m \xrightarrow{*} m'$.*

In the example net of Figure 2.1 the current marking is $p_1 + 2 \cdot p_2 + p_3 + p_4$ and the transition $t_1$ is enabled. If $t_1$ fires the successor marking is given by $p_1 + p_2 + 2 \cdot p_4 + p_5$. From this marking the sequence $t_2 \cdot t_1 \cdot t_2$ is activated resulting in the marking $p_1 + p_3 + 3 \cdot p_4$ in which now no transition is activated, i.e. we have reached a *deadlock*.

Note that the number of tokens on $p_1$ is not changed by firing $t_1$, but that $p_1$ needs to be marked anyway or otherwise $t_1$ is not active and can not fire. Also note that the overall number of tokens remains the same. This does not have to be the case. By deleting the arc $(t_1, p_4)$ for example, this effect would be gone.

In the example of Figure 2.1 there is only one transition active at each reachable marking. Usually this is not the case, too. By reversing the arcs $(p_5, t_2)$ and $(t_2, p_3)$ we end up with a net in which the transitions $t_1$ and $t_2$ are both active in the initial marking $p_1 + 2 \cdot p_2 + p_3 + p_4$. If $t_2$ fires now, $t_1$ is not active anymore, since the place $p_3$ is empty then.

We are now able to define reachability of a marking and liveness of a net, two basic terms in verification.

**Definition 2.32 (Reachability).** *Let $\mathcal{N} = (P, T, \mathbf{pre}, \mathbf{post}, m_0)$ be a p/t net system and $m, m'$ markings of $\mathcal{N}$. The marking $m'$ is* reachable *from the marking $m$ iff a sequence $w \in T^*$ of transitions exists such that $m \xrightarrow{w} m'$ holds, i.e. $w$ is enabled at $m$ and after firing $w$ the current marking of $\mathcal{N}$ is $m'$.*

*The* set of markings reachable from $m \in \mathcal{M}_N$ in $\mathcal{N}$*, denoted by $R(N, m)$, is defined as*

$$R(N, m) := \{m' \in \mathcal{M} \mid \exists w \in T^* : m \xrightarrow{w} m'\}$$

*The* set of reachable markings of $\mathcal{N}$*, denoted by $R(\mathcal{N})$, is the set of markings reachable from $m_0$, i.e. $R(\mathcal{N}) := R(N, m_0)$.*

*The* reachability graph *of $\mathcal{N}$, denoted by $RG(\mathcal{N})$, is a directed graph $G = (V, E)$ with*

$$\begin{aligned} V &:= R(\mathcal{N}) \\ E &:= \{(m, t, m') \in \mathcal{M} \times T \times \mathcal{M} \mid m \xrightarrow{t} m'\}. \end{aligned}$$

*The set of nodes are thus exactly the reachable markings and a labelled edge connects a marking $m$ to a marking $m'$ reachable from $m$ by firing one transition.*

Note that the reachability graph can also be seen as a special transition systems (see Definition 2.15) where the transition relation additionally includes the firing transition (of the Petri net) and the labelling map is not used.

Boundedness can be defined via finiteness of the reachability set.

**Definition 2.33** (**Boundedness**). *Let $\mathcal{N} = (P, T, \mathbf{pre}, \mathbf{post}, m_0)$ be a p/t net system. $\mathcal{N}$ is* bounded *iff the reachability set is finite, i.e. if $|R(\mathcal{N})| < \infty$.*

The following theorem gives an alternative characterisation of boundedness.

**Theorem 2.34.** *Let $\mathcal{N} = (P, T, \mathbf{pre}, \mathbf{post}, m_0)$ be a p/t net system. $\mathcal{N}$ is* bounded*, i.e. $|R(\mathcal{N})| < \infty$, iff a $k \in \mathbb{N}$ exists such that $m(p) \leq k$ for all $m \in R(\mathcal{N})$ and for all $p \in P$, i.e. on each place at most $k$ tokens reside in each reachable marking.*

*Proof.* Let $\mathcal{N}$ be bounded. Among the finite set of markings and the finite set of places there is one marking $m$ that assigns the most tokens to one of the places $p$, i.e. $m(p)$ is maximal. Let $k := \max\{m(p) \mid m \in R(\mathcal{N}), p \in P\}$, then on each place at most $k$ tokens reside in each reachable marking.

Conversely, let $m(p) \leq k$ for some $k$ and all $m \in R(\mathcal{N})$ and $p \in P$. In each marking each place is then assigned a number between 0 and $k$ culminating in at most $(k+1)^{|P|}$ markings, thus $|R(\mathcal{N})| < \infty$. □

**Definition 2.35** (**Liveness**). *Let $\mathcal{N} = (P, T, \mathbf{pre}, \mathbf{post}, m_0)$ be a p/t net system. A transition $t \in T$ is* live *iff for each $m \in R(\mathcal{N})$ a sequence $w \in T^*$ exists that enables $t$, i.e.*

$$t \in T \text{ is live} \quad iff \quad \forall m \in R(\mathcal{N}) \exists w \in T^* m \xrightarrow{wt}$$

*It is thus always possible to enable $t$ again.*

*The p/t net system $\mathcal{N}$ is* live *if every transition $t \in T$ is live.*

We now define standard problems in the context of Petri nets and verification in general. The reachability problem asks if a given marking is reachable in a net system. The liveness problem asks if every transition can always be activated again. The coverability problem asks if a marking $m$ can at least be "covered" by a reachable marking $m'$, i.e. if $m'$ with $m' \geq m$ is reachable. The boundedness problem asks if the set of reachable markings is finite.

**Definition 2.36** (**Petri net problems**).

1. *In the* reachability problem *a p/t net system $\mathcal{N} = (N, m_0)$ and a marking $m$ of $\mathcal{N}$ are given and the question is, if $m_0 \xrightarrow{*} m$ holds.*

2. *In the* liveness problem *a p/t net system $\mathcal{N} = (N, m_0)$ is given and the question is, if $\mathcal{N}$ is live.*

3. *In the* coverability problem *a p/t net system $\mathcal{N} = (N, m_0)$ and a marking $m$ of $\mathcal{N}$ are given and the question is, if a marking $m'$ exists such that $m' \in R(\mathcal{N})$ and $m' \geq m$.*

4. *In the* boundedness problem *a p/t net system $\mathcal{N} = (N, m_0)$ is given and the question is, if $\mathcal{N}$ is bounded.*

All the problems mentioned above are decidable for p/t nets, but to solve them in the general case requires at least exponential space and thus the complexity is rather high. Indeed most interesting questions about p/t nets are ExpSpace-hard (cf. the survey articles by Esparza [Esp98a] and Esparza and Nielsen [EN94]). Note that ExpSpace-hardness does not mean that these problems are decidable. Indeed many important problems for p/t nets are actually undecidable, among them the model checking problems for LTL and CTL and equivalence problems (cf. [Esp98a]).

The reachability problem was first proven to be decidable by Mayr in 1981 [May81] (see also [May84]). A simplified proof was given by Kosaraju in [Kos82]. Unfortunately, the procedures given in both require non primitive recursive space. A further simplification by Lambert [Lam92] is still non primitive recursive. As a lower bound ExpSpace-hardness was proven by Lipton [Lip76] (see also [Esp98a] for a proof using terminology from Petri nets). Neither an upper bound close to this hardness result nor a better lower bound are currently known.

The liveness problem is recursively equivalent to the reachability problem (cf. [Hac74], [Hac76a]) and is thus decidable, but it is again ExpSpace-hard due to [Lip76] (see also [Esp98a]).

The coverability problem is ExpSpace-hard [CLM76] and can be decided by a technique proposed by Karp and Miller in [KM69]. Unfortunately, this technique is non-primitive recursive [Rac78], but also in [Rac78] Rackoff showed that the problem can actually be decided in $2^{c \cdot n \cdot \log n}$ space for some constant $c$ and thus in ExpSpace and with this gave an almost optimal bound.

Decidability of the boundedness problem was also proven by Karp and Miller in [KM69]. Lipton proved that at least $2^{c \cdot \sqrt{n}}$ space is required and thus the problem is ExpSpace-hard [Lip76]. In [Rac78] Rackoff gave an algorithm that again requires $2^{c \cdot n \cdot \log n}$ space and that is thus almost optimal, too.

**Theorem 2.37.** *The reachability, liveness, coverability, and boundedness problem are decidable for p/t nets, but are* ExpSpace*-hard.*

*Coverability and boundedness can be decided in* ExpSpace*.*

On the one hand Petri nets lack certain modelling capabilities. For this reason several extensions are known, most of which yield a Turing complete formalism. We state two prominent ones below, which we will need later on.

On the other hand, the above complexity bounds render the formalism practically intractable with verification in mind. Many restrictions of the formalism have been investigated in which important problems like reachability and liveness are not only decidable, but in which far lesser resources are needed in the decision procedures. These restrictions can be roughly divided into *structural* and *dynamic* restrictions. Some of these restrictions are treated in Section 2.5 below.

## Extension of Petri Nets

For certain modelling tasks Petri nets lack appropriate modelling capabilities. For example, in standard Petri nets it is not possible to test if a certain place has *zero* tokens on it. We state two prominent ones, inhibitor nets and coloured Petri nets, below, which we will need later on.

### Inhibitor Nets

Inhibitor nets are defined as standard Petri nets with an additional set $I$ of arcs called *inhibitor arcs* (see [Hac76c] and [Pet81]). Inhibitor arcs connect places to transitions, i.e. $I \subseteq P \times T$. If $(p,t) \in I$ is an inhibitor arc, then $t$ is only allowed to fire in a marking $m$ if $m(p) = 0$ holds, that is inhibitor arcs test places for zero. Unsurprisingly nets with this capability are Turing-complete, which was shown by Hack by a simulation of counter programs [Hac76c].

Note that inhibitor arcs are defined here as elements of $P \times T$, but their direction is actually not important. They are usually depicted as undirected edges.

**Definition 2.38 (Inhibitor Nets).** *An inhibitor net is a tuple $N = (P, T, F, W, I)$ where $(P, T, F, W)$ is a p/t net and $I \subseteq P \times T$ is a set of inhibitor arcs. A transition $t \in T$ is active in a marking $m : P \to \mathbb{N}$ iff for all $p$ such that an input arc $(p,t) \in F$ exists $m(p) \geq W(p,t)$ holds and for all $p$ such that an inhibitor arc $(p,t) \in I$ exists $m(p) = 0$ holds. The successor marking is defined as in standard p/t nets.*

*For simplicity an arc $(p,t) \in F$ is only allowed if $(p,t) \notin I$ - otherwise $t$ would never be able to fire. Note that $(p,t) \in I$ does not exclude $(t,p) \in F$. In this case $t$ might fire, if $p$ is empty and after firing $p$ is marked and must at first be depleted before $t$ can be activated again.*

**Theorem 2.39.** *Inhibitor nets are Turing-complete. Among other problems the reachability problem is undecidable.*

Petri nets with only *one* inhibitor arc are not Turing-complete. The reachability problem can be decided for them, cf. [Rei08].

### Coloured Petri Nets

*Coloured Petri nets* (CPNs) are a widely used extension of classical Petri nets. Tokens in CPNs are not only anonymous black tokens anymore, but may be given any of a specified set of *colours*. In this way it is possible to define complex data. Note, however, that it is not comfortably possible to model internal behaviour of these objects.

Coloured Petri nets may furthermore use *variables* and so called *guards* that are just logical expressions mapped to a transition. The transition is then

only able to fire, if the expression is satisfied. If the set of colours is infinite CPNs are Turing-complete. If only finite set of colours are allowed, a CPN can be *unfolded* into a p/t net and CPNs with finite sets of colours are thus no more expressive than standard Petri nets, albeit modelling might be much more comfortable.

We do not give a formal definition of CPNs here. A detailed account of them including modelling and applications can be found in the books by Jensen [Jen97a], [Jen97b], [Jen97c] and in the newer book by Jensen and Kristensen [JK09].

An interesting extension of CPNs is the addition of *channels for synchronous communication* introduced by Christensen and Hansen [CH94]. With these channels a transition may "send" a token from one of its input places to an output place of another transition. This idea is similar to the idea of transporting net tokens in object net systems (see Chapter 6. However, even with this extension it is still not possible to model an internal structure or behaviour of the tokens of the CPN. See also the discussion of related formalisms in Chapter 3.4.

## 2.5 Restrictions of Petri Nets

Due to the complexity of many important problems for general Petri nets, automatically verifying them is practically intractable (see Theorem 2.37). Therefore many restrictions of the formalism have been introduced and investigated in which important problems like reachability and liveness become practically decidable, i.e. decidable using an affordable amount of resources. These restrictions can be roughly divided into *structural* and *dynamic* restrictions.

### Structural Restrictions of Petri Nets

Given a p/t net system $\mathcal{N} = (P, T, F, m_0)$ a structural restriction somehow restricts the (graph-theoretic) structure of the underlying net $N = (P, T, F)$ by e.g. restricting the number of outgoing arcs from a place.

In the following we treat the restricted net classes we will need later on in the context of object net systems. We will define the classes and state results concerning most importantly the reachability problem and in some cases also results concerning some of the other problems from Definition 2.36.

Not extensively treated here are mixtures of this structural restrictions like e.g. 1-conservative *and* free-choice p/t nets.

Also note that many more structural restrictions are imaginable. See e.g. [EN94] for some more restrictions and also more problems than those defined above.

### P- and T-nets

One of the most severe restriction of a p/t net is that to P- and T-nets. In a P-net the transitions are roughly speaking identical to the arcs, so only the places and arcs are of importance, thus the name P-net. In a T-net the analogous case holds for places.

**Definition 2.40 (P- and T-nets).** *Let $N = (P, T, F)$ be a p/t net.*

1. *$N$ is a* P-net*, if $|{}^\bullet t| = |t^\bullet| = 1$ holds for all $t \in T$.*
2. *$N$ is a* T-net*, if $|{}^\bullet p| = |p^\bullet| = 1$ holds for all $p \in P$.*

*P-nets are also called* S-nets *and T-nets are also called* marked graphs *for historical reasons.*

*If $N$ is a P-net and additionally a initial marking $m_0 \in \mathcal{M}_N$ is given, then $(N, m_0)$ is called a* P-system*. Analogous $(N, m_0)$ is a* T-system *if $N$ is a T-net.*

In a P-net synchronisations are ruled out. In a T-net conflicts can not appear.

Given this severely restricted structure it is not surprising that most problems can be decided very quickly:

**Theorem 2.41.** *For P- and T-systems the reachability and the liveness problem can be decided in polynomial time.*

A proof of the above theorem and of some more properties of P- and T-nets can be found in [DE95], while the origins of this work can be traced back to [CHEP71], [GL73], and [BT87].

### Conservative Petri Nets

Conservative Petri nets are similar to P-nets, but slightly more liberal.

**Definition 2.42 (1-conservative Petri nets).** *A p/t net $N = (P, T, F)$ is* 1-conservative *if $|{}^\bullet t| = |t^\bullet|$ for all $t \in T$.*

*If $N$ is a 1-conservative net, then $(N, m_0)$ with $m_0 \in \mathcal{M}_N$ is a* 1-conservative system*.*

The more general definition of conservative Petri nets requires a function $f : P \to \mathbb{N} \setminus \{0\}$ such that $\sum_{p \in {}^\bullet t} f(p) = \sum_{p \in t^\bullet} f(p)$ for every transition $t$.

We do not need this generalised definition here and focus on 1-conservative Petri nets.

Despite their similarity to P-nets, where many interesting problems can be decided in polynomial time, problems like reachability and liveness become surprisingly hard:

**Theorem 2.43.** *For* 1*-conservative Petri nets the reachability and the coverability problem are* PSPACE*-complete. The liveness problem for* 1*-conservative Petri nets is in* PSPACE*.*

Proofs for the statements in Theorem 2.43 can be found in [JLL77].

1-conservative Petri nets are by definition bounded, because a transition removes as many tokens as it produces and thus the number of tokens remains constant.

### Acyclic Petri nets

A p/t net is acyclic if it does not contain a cycle, i.e. there is no element $x \in P \cup T$ such that there is a non-trivial path starting and ending at $x$.

**Definition 2.44** (**Acyclic Petri nets**)**.** *Let $N = (P, T, F)$ be a p/t net. $N$ is* acyclic *if there is no $x \in P \cup T$ such that $(x, x) \in F^+$ holds.*

*If $N$ is an acyclic net, then $(N, m_0)$ with $m_0 \in \mathcal{M}_N$ is an* acyclic system*.*

Acyclic Petri nets are rarely used due to their restricted modelling power. That the reachability problem for them can be solved in nondeterministic polynomial time follows from a reduction to the linear programming problem for integers [CEP93]. The last mentioned problem is in NP [HU79]. NP-hardness was proved by Stewart [Ste95] along with other results concerning even more restricted classes of acyclic Petri nets. Most notably it was shown there that the reachability problem for 1-safe acyclic Petri nets remains NP-complete (see the discussion about dynamic restrictions below).

**Theorem 2.45.** *The reachability problem for acyclic Petri nets is* NP*-complete.*

### Conflict-free Petri Nets

The definition of conflict-freedom ensures that whenever two transitions are in conflict, i.e. both transitions are activated and have at least one place in their presets in common, they actually both may fire and thus there is not truly a conflict.

**Definition 2.46** (**Conflict-free Petri nets**)**.** *Let $N = (P, T, F)$ be a p/t net. $N$ is* conflict-free *if $p^\bullet \subseteq {}^\bullet p$ holds for each place $p$ with $|p^\bullet| > 1$.*

*If $N$ is a conflict-free net, then $(N, m_0)$ with $m_0 \in \mathcal{M}_N$ is a* conflict-free system*.*

A p/t net in which two activated transitions may both fire without disabling the other is called *persistent* (see the subsection about dynamic restrictions below) and it is easy to come up with a persistent p/t net that is not conflict-free in the sense of the definition above. Nevertheless, the structural definition

covers a nice (sub-)class of p/t nets and can be easily checked. Moreover, conflict-free p/t nets have the nice property that many problems are far easier to solve as in the case of general p/t nets.

**Theorem 2.47.** *The boundedness and the liveness problem for conflict-free p/t nets can be solved in polynomial time. The reachability problem for conflict-free p/t nets is* NP*-complete.*

*Proof.* A proof that boundedness for conflict-free p/t nets can be solved in polynomial time can be found in [HRY87]. The proof for liveness can be found in [HR89] and the proof of NP-completeness of the reachability problem is in [JLL77] and [HR88]. □

The reachability problem can even be solved in polynomial time, if the conflict-free p/t net is also bounded (cf. [HR89] and also see the discussion of dynamic restrictions below).

### Free-Choice Petri Nets

Free-choice Petri nets constitute one of the most thoroughly studied net classes and they are widely reckoned as a large net class with a sensible trade-off between expressiveness and the existence of efficient algorithms. It is stated in [DE95] that Eike Best coined the term "free-choice hiatus" in 1986 to describe the situation that a rich and useful theory for free-choice Petri nets exists, but that few of the results can be extended to larger net classes, i.e. the border between Petri net formalisms that have acceptable expressiveness *and* efficient algorithms for analysis and Petri net formalisms that may be suited to model a richer variety of systems, but for whom decision procedures are inefficient can be drawn very close to free-choice Petri nets.

While in a P-net no synchronisations and in a T-net no conflicts can appear, they are both allowed in a free-choice net, but interference of both is ruled out. Free-choice Petri nets have been introduced in [Hac72]. See also the thorough treatment in [DE95].

**Definition 2.48** (**Free-choice Petri nets**)**.** *Let $N = (P, T, F)$ be a p/t net. $N$ is* free-choice *if $(p, t) \in F$ implies ${}^\bullet t \times p^\bullet \subseteq F$ for every place $p$ and every transition $t$ or alternatively if for every two places $p_1, p_2$ and two transitions $t_1, t_2$ $(p_1, t_1), (p_1, t_2), (p_2, t_1) \in F$ implies $(p_2, t_2) \in F$.*

*If $N$ is a free-choice net, then $(N, m_0)$ with $m_0 \in \mathcal{M}_N$ is a* free-choice system*.*

Free-choice Petri nets have been subject to extensive research. Theorem 2.49 lists some of the most important results concerning the reachability and the liveness problem. Most results require additional dynamic restrictions like liveness or boundedness. 1-safeness means that at every reachable marking and on each place at most one black token resides (see also Definition 2.50 below). A

net is *cyclic* if the initial marking can be reached from any reachable marking again.

**Theorem 2.49.** *1. Reachability is* PSPACE*-complete for* 1*-safe free-choice p/t nets.*

*2. Reachability is* NP*-complete for live and bounded free-choice p/t nets.*

*3. Reachability is solvable in polynomial time for live, bounded, and cyclic free-choice p/t nets.*

*4. Liveness is* coNP*-complete for free-choice nets.*

*5. Liveness is solvable in polynomial time for bounded free-choice nets.*

A proof outline for the first item can be found in [CEP95], the full construction is in [JLL77] and [Hac76b], see also the subsection about dynamic restrictions below. The second item is proved in [Esp98b]. The third item is proved in [DE91], [DE93]. It can also be decided in polynomial time if these three properties hold for certain nets [DE91] and if a free-choice net is live and bounded [DE95].

The fourth item is proved in [JLL77], see also [DE95] and the fifth item is proved in [ES92].

A thorough treatment of free-choice Petri nets can be found in the text book by Desel and Esparza [DE95], which only lacks newer result like [Esp98b] that proves the second item above.

## Dynamic Restrictions of Petri Nets

Contrary to structural restrictions a dynamic restriction of a p/t net system does not restrict the structure of the underlying net, but somehow takes into account the dynamic of the system, i.e. the firing of transitions or the markings reachable by firing of transitions. For example demanding that a given net system is live is a dynamic restrictions because regardless of the marking reached for every transition $t$ there is always a sequence of transitions whose firing enables $t$ again.

In the following we will again treat the restricted net classes we will need later on in the context of object net systems. We will define the classes and state results concerning the problems from Definition 2.36.

Some more dynamic restrictions can be found in [EN94].

### Unary and Persistent Petri Nets

A Petri net is *unary* if in every reachable marking at most one transition is enabled. This severely restricts the possible reachability graphs. But even

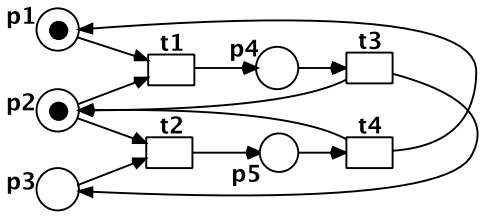


Figure 2.3: A persistent but not conflict-free p/t net

in this case the reachability problem is PSPACE-hard. Indeed, it is PSPACE-complete in the case of unary and 1-safe Petri nets (cf. [CEP95]).

A Petri net is *persistent* if whenever two transitions are enabled at a reached marking, then both may fire and in particular one does not disable the other. Note that a unary net is persistent and that a conflict-free net is also persistent. Indeed, persistence tries to capture the absence of conflicts in a dynamic way, but it is hard to check if a net is persistent. For example it is PSPACE-complete to decide if a net is persistent even for 1-safe nets [CEP95]. While a conflict-free net is also persistent, it is easy to come up with a persistent net that does not have the structural property to be conflict-free (see Figure 2.3 for an example). Thus conflict-freedom might not capture all nets in which no conflicts arise, but it is an easy to check structural property, it covers a wide range of the nets without conflicts, and many problems become quickly decidable as stated in the discussion of conflict-free Petri nets above.

### Safe Petri Nets

Safeness restricts the number of tokens that may reside on a place. In a safe net all places may only contain a limited number of tokens and thus the state space is finite. Also if the state space is finite, the net is $k$-safe for some $k$.

**Definition 2.50** (**$k$-safe Petri nets**). *Let $\mathcal{N} = (P, T, F, m_0)$ be a p/t net system. $\mathcal{N}$ is* $k$-safe, *$k \in \mathbb{N}$, if for all places $p \in P$ and all reachable markings $m \in R(\mathcal{N})$ the number of tokens on $p$ is bounded by $k$, i.e. $\forall m \in R(\mathcal{N}) \forall p \in P : m(p) \leq k$.*

*A p/t net that is $n$-safe for some $n$ is also called bounded.*

*If $\mathcal{N}$ is 1-safe, we simply say that $\mathcal{N}$ is* safe.

**Lemma 2.51** (**Finiteness of $k$-safe Petri nets**). *A p/t net $\mathcal{N}$ is $k$-safe for some $k$ iff $|R(\mathcal{N})| < \infty$, i.e. the state space is finite.*

*Proof.* Let $n := |P|$. Since at each of the $n$ places between 0 and $k$ tokens reside, there are at most $(k+1)^n$ markings of $N$.

For the other direction let $|R(\mathcal{N})| < \infty$. Let $k := \max\{m(p) \mid p \in P, m \in R(\mathcal{N})\}$, then $\mathcal{N}$ is $k$-safe, because by definition of $k$ in no reachable marking more than $k$ tokens reside on any place of $\mathcal{N}$. □

Finiteness of state-space does not only guarantee decidability of interesting problems, many problems become far quicker decidable than for general Petri nets.

**Theorem 2.52.** *1. To decide if a given net is* 1*-safe is* PSPACE*-complete.*

*2. Reachability is* PSPACE*-complete for (unary)* 1*-safe p/t nets.*

*3. Liveness is* PSPACE*-complete for* 1*-safe p/t nets.*

*4. Coverability is* PSPACE*-complete for* 1*-safe p/t nets.*

*5. Every property that can be expressed in the temporal logics CTL or LTL can be decided deterministically in polynomial space for* 1*-safe p/t nets.*

May of the results above are mentioned or proved in [CEP95], but actually follow from the fact already observed in the late seventies by Jones, Landweber, and Lien in [JLL77], that a 1-safe p/t net of size $O(n^2)$ can simulate a linear bounded automaton starting on an empty tape of size $n$. Since the net can be constructed in polynomial time, hardness results concerning linear bounded automata carry over to 1-safe p/t nets. Proofs for the second, third, and fourth item of Theorem 2.52 can be found in [CEP95]. For reachability the statement is also true for *unary* 1-safe p/t nets, where unary means that at most one transition is enabled at every reachable marking. Boundedness is always true for 1-safe or bounded nets.

The first item of Theorem 2.52 is proved in [JLL77].

The strong fifth item is proved in [Esp98a].

Safeness is especially interesting in combination with structural restrictions. For net classes whose nets are 1-safe and also have some structural restriction imposed upon them like e.g. conflict-freedom or free-choice, the reachability problem might be even quicker solvable. This line of research has been intensively studied and we state results relevant for this thesis in Theorem 2.53.

Some of the results of Theorem 2.53 have already been stated in the preceding sections. For convenience we group them together here anyway.

**Theorem 2.53.** *1. Reachability for* 1*-safe and acyclic p/t nets is* NP*-complete.*

*2. Liveness for* 1*-safe and acyclic p/t nets can be solved in constant time.*

*3. Reachability for* 1*-safe and conflict-free p/t nets is in* P*.*

*4. Liveness for* 1*-safe and conflict-free p/t nets is in* P*.*

5. *Reachability for* 1*-safe and free-choice p/t nets is* PSPACE*-complete.*

6. *Liveness for* 1*-safe and free-choice p/t nets is in* P*.*

7. *Reachability for* 1*-conservative p/t nets is* PSPACE*-complete.*

8. *Liveness for* 1*-conservative p/t nets is in* PSPACE*.*

Stewart showed in [Ste95] the first item. The second item is mentioned in [CEP95].

Both results for safe and conflict-free p/t nets can be found in [HR89]. Note the difference to the unbounded case in which the reachability problem is NP-complete (see Theorem 2.47).

A proof outline for the fifth item can be found in [CEP95] the full construction is in [JLL77] and [Hac76b]. The sixth item is proved in [ES92]. See also the subsection about free-choice p/t nets above for more results on free-choice nets.

Item seven and eight are proved in [JLL77].

# 3 Elementary Object Systems

In this chapter, we recapitulate *Elementary Object Systems* (Eos), a nets-within-nets-formalism introduced by Valk [Val91] in which the nesting is restricted to two levels, i.e. there is a p/t net, called the *system net*, in which the tokens are allowed to be standard p/t nets, called *object nets*. We give a small example in this chapter and point to the literature for a case study showing the usefulness of the approach for modelling agent systems. On the negative side we show that even with the restriction of the nesting depth to two levels, elementary object systems are Turing-powerful, thus preventing attempts to find usable analysis techniques.

In Section 3.1, we motivate the formalism of Eos and explain it in an intuitive way with an example. In Section 3.2, we give the formal definition of Eos. Our presentation in this section is close to the presentation in the report by Köhler-Bußmeier [KB11], which summarises most of the work on Eos before this thesis. In Section 3.3, we prove in two different ways that Eos are as powerful as Turing machines, which implies undecidability of the reachability problem and other problems. Firstly, we recapitulate, with slight modifications, the proof from Köhler in [Köh07], in which a simulation of counter programs is presented. Secondly, we present a construction to simulate inhibitor nets by Eos providing an alternative proof that is reusable later in the context of conservative Eos (see Section 4.1). The second construction is joint work with Michael Köhler-Bußmeier and is published in [KBH11a] and in revised form in [KBH12].

In Section 3.4, we discuss other approaches which also try to capture the notion of both nesting of structures and mobility of objects. Among these approaches, some are building upon the theory of Petri nets, for example nested nets [Lom00] or hypernets [BBPP05], and some originate from process algebra, for example the Ambient Calculus [CG00b] and the Seal Calculus [CVN05].

## 3.1 Motivation

Imagine a robot that moves around in some environment such as a factory. The robot picks up some materials from the environment, say from the storage depot, and transports it to some machine, where the materials are used to construct an artefact. A Petri net modelling the environment might look like

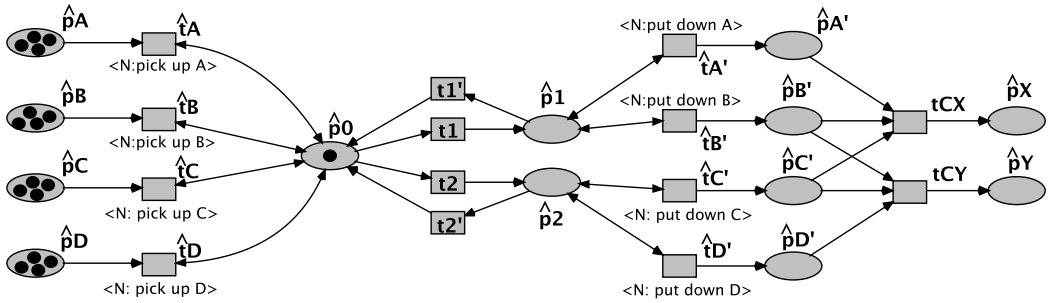


Figure 3.1: The environment modelled by a Petri net $\widehat{N}$.

the one depicted in Figure 3.1. For now ignore the transitions' labelings in brackets.

Initially the robot resides on the place $\widehat{p}_0$. He can then pick up the different materials $A$, $B$, $C$, or $D$ with the transitions to the left. The materials are only available in limited quantity. The robot can also move to the places $\widehat{p}_1$ or $\widehat{p}_2$ which model workbenches, there it can put down the materials, picked up before. The materials can then be used via the two rightmost transitions to construct the artefacts $X$ or $Y$. Construction is only possible if the robot has delivered the right materials. That the robot may only reside on the places $\widehat{p}_0, \widehat{p}_1$, and $\widehat{p}_2$ is modelled by the typing (not depicted in the figures). These places are thus explicitly typed with the object net that models the robot. All other places are typed with the object net type for black tokens.

The requirement, that only materials that have been picked up before, can be put down, is not yet modelled. Intuitively, this should be encoded *in the robot*. Of course, in this case we would also be able to model the whole scenario with a standard Petri net. It is far more natural, however, to model the robot with something more expressive than a black token. In the nets-within-nets-approach the black token can be a Petri net again and thus it is possible to model internal structure and internal behaviour of the object modelled before with only the black token.

To model the robot with a Petri net, we note that the robot can only pick up and put down certain objects. A Petri net, modelling this behaviour and remembering which objects have been picked up, is depicted in Figure 3.2.

The net $\widehat{N}$ in Figure 3.1, modelling the environment, is called the *system net*. The net $N$ in Figure 3.2, modelling the robot or agent, is called an *object net*. The transitions' inscriptions `pick up A` etc. are *channels*. Channels are used in the following way: In the system net the inscription `<N:pick up A>` of the transition $\widehat{t}_A$ means that to be able to fire, $\widehat{t}_A$ needs to *synchronise* with a transition of the object net $N$, that is labelled with the same channel `pick up A`. In the object net the inscription `<:pick up A>` of the transition $t_A$ means similarly that to be able to fire, $t_A$ needs to synchronise with a transition of

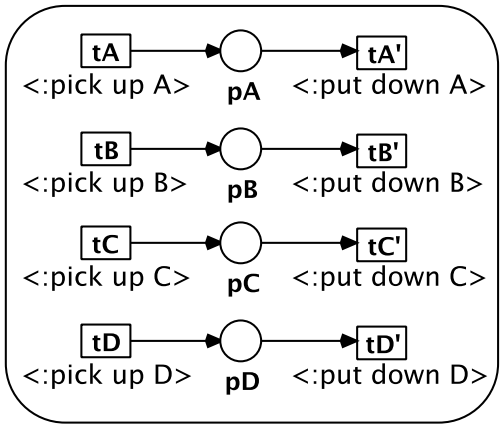


Figure 3.2: The robot modelled by a Petri net $N$ and located on place $\widehat{p}_0$ of Figure 3.1.

the system net, that is labelled with the same channel. Since there is only one system net, it is not necessary to explicitly state it in the object nets' transition inscriptions. The transition $\widehat{t}_A$ of the system net may thus only fire synchronously with $t_A$ in the object net.

The places $\widehat{p}_0, \widehat{p}_1$, and $\widehat{p}_2$ are typed with $N$, meaning that on these places only net tokens of this object net type $N$ may reside. All other system net places are typed with $N_\bullet$, meaning that on these places only standard black tokens may reside.

We now describe the dynamic behaviour of the net system. Initially the situation is as depicted in Figure 3.1, where the robot modelled with the net of Figure 3.2 resides on $\widehat{p}_0$. Thus we do not have a black token but a Petri net on $\widehat{p}_0$. The marking of the robot is the empty marking $\mathbf{0}$, since initially it has not picked up any materials. Markings are described via nested multisets. We thus start with the marking

$$\widehat{p}_0[\mathbf{0}] + 4 \cdot \widehat{p}_A + 4 \cdot \widehat{p}_B + 4 \cdot \widehat{p}_C + 4 \cdot \widehat{p}_D,$$

meaning that $\widehat{p}_0$ is marked with a net token whose marking is $\mathbf{0}$ and the places $\widehat{p}_A, \widehat{p}_B, \widehat{p}_C$, and $\widehat{p}_D$ are marked with four black tokens each. That on $\widehat{p}_0$ resides a net token of type $N$ and that black tokens reside on the other places originates from the typing of the places. The synchronous event consisting of $\widehat{t}_A$ and $t_A$, denoted by $\widehat{t}_A[t_A]$, is activated, since both transitions are activated and can thus fire, resulting in the successor marking

$$\widehat{p}_0[p_A] + 3 \cdot \widehat{p}_A + 4 \cdot \widehat{p}_B + 4 \cdot \widehat{p}_C + 4 \cdot \widehat{p}_D.$$

Firing $\widehat{t}_A[t_A]$ once again, $\widehat{t}_B[t_B]$ twice, and $\widehat{t}_C[t_C]$ once leads to

$$\widehat{p}_0[2 \cdot p_A + 2 \cdot p_B + 1 \cdot p_C] + 2 \cdot \widehat{p}_A + 2 \cdot \widehat{p}_B + 3 \cdot \widehat{p}_C + 4 \cdot \widehat{p}_D.$$

Then we fire the *system-autonomous* event $t_1$ which models a *movement* of the robot to $\widehat{p}_1$. There the robot puts down its materials from $\widehat{p}_A$ and $\widehat{p}_B$ by firing the synchronous events $\widehat{t}'_A[t'_A]$ (twice) and $\widehat{t}'_B[t'_B]$ (twice). Note that due to the typing of the places, the object net $N$ is placed on $\widehat{p}_1$ again and not on $\widehat{p}'_A$ or $\widehat{p}'_B$. The places $\widehat{p}'_A$ and $\widehat{p}'_B$ are marked with black tokens.

The material originally obtained from $\widehat{p}_C$ can not be dropped from $\widehat{p}_1$. The robot thus moves via the system-autonomous event $t'_1$ back to $\widehat{p}_0$ and then via $t_2$ to $\widehat{p}_2$. There the robot drops its last material via $\widehat{t}'_C[t'_C]$, resulting in the marking

$$\widehat{p}_1[\mathbf{0}] + 2 \cdot \widehat{p}_A + 2 \cdot \widehat{p}_B + 3 \cdot \widehat{p}_C + 4 \cdot \widehat{p}_D + 2 \cdot \widehat{p}'_A + 2 \cdot \widehat{p}'_B + 1 \cdot \widehat{p}'_C.$$

The robot can then return to its initial position via $t'_2$ and with $t_{cx}$ one artefact $X$ can be constructed resulting in

$$\widehat{p}_0[\mathbf{0}] + 2 \cdot \widehat{p}_A + 2 \cdot \widehat{p}_B + 3 \cdot \widehat{p}_C + 4 \cdot \widehat{p}_D + 1 \cdot \widehat{p}'_A + 1 \cdot \widehat{p}'_B + 1 \cdot \widehat{p}_X.$$

Many additions to the model are imaginable, for example, extending the model of the robot with a complementary place to allow only a certain amount of materials to be picked up. However, the rough model above is enough to illustrate the main points of the formalism.

In the example above we had synchronous and system-autonomous events. Object-autonomous events are also possible. In this case a transition in an object net, not using any channel, fires autonomously from any system net transition.

A typical verification question now would be if a certain object can be constructed or if the robot can always return to its initial position. Thus being able to solve the reachability problem and maybe other questions, expressible in commonly used logics like CTL or LTL, is of great importance. If the general formalism is Turing-complete, the question is if all facets of it are used in the example above or if the formalism can be restricted in such a way that the models are still useful, but can also be verified.

Note that it might be helpful from the modeller's point of view to have the ability to use colours, instead of black tokens, to combine the different materials in one place, for example. This is not investigated further in this thesis. The focus is on the simpler model above. Furthermore, note that it might also be helpful to model the materials as Petri nets again, in case these are complex objects with internal structure and behaviour again. This is already possible in the outlined formalism above. However, in that case it would be nice to actually transport these objects including their current internal state *in the vertical dimension*, that is from the environment to the robot or more abstract from the system net to the object net. In the example above the object net on place $\widehat{p}_A$ would then be transported to the place $p_A$ in the object net modelling the robot. Since the object net modelling the material then resides in the object

net modelling the robot which again resides in the system net and we thus have three nesting levels, these thoughts lead to object nets with a deeper or even arbitrary nesting. Moreover, these thoughts lead to a formalism that allows such a vertical transport of tokens as mentioned above. This topic is taken up in detail in Chapter 6. In this chapter, we now focus on the formal definition of the basic model, introduced informally above and on the expressiveness of this formalism. Another example of the usage of object net systems can be found in [KMR03], where they are used in the context of agent systems and mobile robots.

In the following we will speak of the *nets-within-nets-family* of formalisms if we want to denote a formalism which somehow applies the idea of nesting to Petri nets (see the discussion of Petri net like formalisms in Section 3.4).

The term *object net system* is used to emphasise that we mean a member of the nets-within-nets-family, where the black tokens of a Petri net can be a Petri net again or a reference to a Petri net. In particular the formalisms introduced in this thesis are object net systems. The term is also used for the most general form we introduce in Chapter 6.

## 3.2 Fundamentals and Formal Definition of Eos

In the following we formally define the formalism of *elementary object systems* introduced intuitively in the previous section.

Eos where first introduced in [Val91], while our definitions follow [KB11] with slight modifications. Most notably, in our definition the labelling $l = (\widehat{l}, (l_N)_{N\in\mathcal{N}})$ is part of the tuple of an Eos instead of the set of events $\Theta$ (see Definition 3.1 below). This is more natural from a modelling point of view and also more amenable to complexity theoretic analysis, because $\Theta$ tends to be very big (cf. Lemma 3.10) and thus statements about the complexity of some procedure *in the size of the object net system* would not be helpful if $\Theta$ would be part of the object net system.

As informally described in the previous section, an elementary object system is composed of a system net, a set of object nets, a typing of the system net places and a labelling of the transitions with channels or with the special symbol $\tau$. The labelling has to adhere to some restrictions and from the labelling the set of events can be deduced. An example is given in Figure 3.3 below, see also Example 3.5 below.

**Definition 3.1** (**Eos**). *Let $C$ be a set of* synchronisation channels *and $\tau \notin C$. An* elementary object system *(*Eos *for short) is a tuple $OS = (\widehat{N}, \mathcal{N}, d, l)$ such that:*

1. $\widehat{N} = (\widehat{P}, \widehat{T}, \mathbf{pre}, \mathbf{post})$ *is a p/t net, called the* system net.

2. $\mathcal{N} = \{N_1, \ldots, N_n\}$ *is a finite set of disjoint p/t nets called* object nets *given as* $N_i = (P_{N_i}, T_{N_i}, \mathbf{pre}_{N_i}, \mathbf{post}_{N_i})$.

3. $d : \widehat{P} \to \mathcal{N}$ *is the* typing *of the system net places.*

4. $l = (\widehat{l}, (l_N)_{N \in \mathcal{N}})$ *is the* labelling *in which*

$$\widehat{l} : \widehat{T} \to (\mathcal{N} \to (C \cup \{\tau\})) \text{ and}$$
$$l_N : T_N \to (C \cup \{\tau\}) \text{ for all } N \in \mathcal{N}.$$

*All these functions are total.*

We assume $\widehat{N} \notin \mathcal{N}$ and the existence of the object net $N_\bullet \in \mathcal{N}$ which has no places or transitions and is used to model black tokens. Moreover, we assume that all sets of nodes (places and transitions) are pairwise disjoint and set $P_{\mathcal{N}} := \cup_{N \in \mathcal{N}} P_N$ and $T_{\mathcal{N}} := \cup_{N \in \mathcal{N}} T_N$.

The system net places are typed by the mapping $d : \widehat{P} \to \mathcal{N}$ with the meaning, that if $d(\widehat{p}) = N$, then the place $\widehat{p}$ of the system net may contain only net-tokens of the object net type $N$.

The transitions in an EOS are labelled with synchronisation channels by the synchronisation labelling $l$. For this we assume a fixed set of channels $C$. In addition we allow the label $\tau$ which is used to describe that no synchronisation is desired (i.e. autonomous firing). The intended meaning of the labels is as follows:

1. $l_N(t) = \tau$ means that the transition $t$ of the object net $N$ may fire *object-autonomously* or simple *autonomously.*

2. $l_N(t) = c \neq \tau$ means that $t$ synchronises via the channel $c$ with the system net.

3. $\widehat{l}(\widehat{t})(N) = \tau$ means that the system net transition $\widehat{t}$ may fire independently (or autonomously) from the object net $N$. If $\widehat{l}(\widehat{t})(N) = \tau$ holds for all $N \in \mathcal{N}$ then the system net transition $\widehat{t}$ may fire *system-autonomously* or simply *autonomously.*

4. $\widehat{l}(\widehat{t})(N) = c \neq \tau$ means that $\widehat{t}$ synchronises via the channel $c$ with the object net $N$.

In case of a autonomous event a single transition fires independently from all other transitions. In case of a synchronous event a system net transition fires together with a non-empty set of object net transitions. In this case the labels, i.e. the channels used, have to match (see Definition 3.4 below).

**Definition 3.2** (**Marking of an Eos**). *A* marking *of an* Eos *OS is a* nested *multiset. Let*

$$M_{\mathcal{N}} := \bigcup_{\widehat{p} \in \widehat{P}} (\{\widehat{p}\} \times \mathcal{M}_{d(\widehat{p})}).$$

*A marking of OS is a finite multiset* $\mu : M_{\mathcal{N}} \to \mathbb{N}$.

*We will usually denote an element* $(\widehat{p}, M)$ *of* $M_{\mathcal{N}}$ *by* $\widehat{p}[M]$ *and a marking of an* Eos *by* $\mu = \sum_{k=1}^{|\mu|} \widehat{p}_{i_k}[M_{i_k}]$, *where* $\widehat{p}_{i_k}$ *is a place in the system net,* $M_{i_k}$ *is a marking of a net token of type* $d(\widehat{p}_k)$, *and the* $i_k$ *are indices. With* $|\mu|$ *we denote the number of elements from* $M_{\mathcal{N}}$ *that appear in* $\mu$, *i.e. the number* $\sum_{(\widehat{p},m) \in M_{\mathcal{N}}} \mu(\widehat{p}, m)$, *which is the number of net tokens present in* $\mu$. *As a shortcut we write the sum also as* $\mu = \sum_k \widehat{p}_k[M_k]$.

*We define the partial order* $\leq$ *on nested multisets by setting* $\mu_1 \leq \mu_2$ *iff* $|\mu_1| \leq |\mu_2|$ *and the elements of* $\mu_1$ *and* $\mu_2$ *can be arranged in such a way that* $\mu_1 = \sum_i \widehat{p}_i[M_i]$, $\mu_2 = \sum_j \widehat{p}'_j[M'_j]$ *and for all* $k \leq |\mu_1|$ *we have* $\widehat{p}_k = \widehat{p}'_k$ *and* $M_k \leq M'_k$, *where* $\leq$ *is the usual multiset relation (see Chapter 2.1). Thus the same net tokens appear in* $\mu_2$ *with at least the same marking as in* $\mu_1$ *and in* $\mu_2$ *may appear additional net tokens.*

*With* $\preceq$ *we denoted the special partial order with* $\mu_1 \preceq \mu_2$ *iff* $|\mu_1| \leq |\mu_2|$ *and the elements of* $\mu_1$ *and* $\mu_2$ *can be arranged in such a way that* $\mu_1 = \sum_i \widehat{p}_i[M_i]$, $\mu_2 = \sum_j \widehat{p}'_j[M'_j]$ *and for all* $k \leq |\mu_1|$ *we have* $\widehat{p}_k = \widehat{p}'_k$ *and* $M_k = M'_k$. *Thus every net token that appears in* $\mu_1$ *also appears in* $\mu_2$ *and with the same marking. In* $\mu_2$ *additional net tokens may appear.*

*The set of all markings of OS is denoted by* $\mathcal{M}_{OS}$ *or simply* $\mathcal{M}$ *if no ambiguity can arise.*

*An* Eos with initial marking *is a tuple* $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *where* $\mu_0 \in \mathcal{M}$ *is the initial marking.*

As was necessary for p/t net systems the size of an Eos has to be defined to state complexity results (see Definition 2.28)

**Definition 3.3** (**Size of an Eos**). *Let* $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *be an* Eos. *The* size *of OS is defined as* $|OS| := |\widehat{N}| + \sum_{N \in \mathcal{N}} |N| + |d| + |l| + |\mu_0|$, *where the size of the p/t nets is defined in Definition 2.28. The typing d can be stored in a one-dimensional array with* $|d| \in O(|\widehat{P}|)$. *The size of the labelling* $|l|$ *is the sum of the space needed to store* $\widehat{l}$ *and all the* $l_N$ *for* $N \in \mathcal{N}$, *where the* $l_N$ *can again be stored in a one-dimensional array each and* $\widehat{l}$ *can be stored in a two-dimensional array. The marking* $\mu_0$ *is stored as an array of* $|\widehat{P}|$ *elements where each element either points to* null, *if the corresponding system net place is unmarked, or to an array of* $|P|$ *elements where* $P$ *is the set of places of the object net type that resides on that system net place.*

We usually do not distinguish between an Eos and an Eos with initial marking and often simply speak of an Eos even if an initial marking is present. What is meant will always be clear from the context.

In the above definition an element of $M_{\mathcal{N}}$ is a marking of an object net together with the place in the system net on which it resides. A marking $\mu$ of *OS* then gives the position of all net tokens and their inner marking. Multiplicity is only needed (in $\mu$) if more than one net token *with the same marking* resides on the same place of the system net.

To introduce the firing rule for Eos we need to introduce *events*, which correspond to transitions in a p/t net, *projections* and the *enabling predicate*. In the following definitions we usually assume that an Eos as in Definition 3.1 is given.

The set of system events $\Theta$ is generated by the synchronisation labelling. The set $\Theta$ consists of the disjoint sets of synchronous events $\Theta_l$, object-autonomous events $\Theta_o$, and system-autonomous events $\Theta_s$. An event is a pair, denoted $\widehat{t}[\vartheta]$ in the following, where $\widehat{t}$ is a transition of the system net or $\widehat{\epsilon}$ and $\vartheta$ maps each object net to one of its transitions or to $\epsilon$. The mapping has to be consistent with the labelling and in an synchronous event the labels of the participating transitions have to match. The symbol $\widehat{\epsilon}$ is used for object-autonomous firing. The special mapping that maps each object net to $\epsilon$, i.e. $\vartheta(N) = \epsilon$ for all $N$ is denoted by $\vartheta_\epsilon$. In such an event the system net transition fires autonomously.

**Definition 3.4** (**Events**). *An event is a pair denoted $\widehat{t}[\vartheta]$ where $\widehat{t}$ is a transition of the system net or the special symbol $\widehat{\epsilon}$ and $\vartheta : \mathcal{N} \to T_{\mathcal{N}} \cup \{\epsilon\}$ is a mapping where $\vartheta(N) \neq \epsilon$ implies $\vartheta(N) \in T_N$ for all $N \in \mathcal{N}$.*

*The labelling functions are extended to $l_N(\epsilon) = \tau$ and $\widehat{l}(\widehat{\epsilon})(N) = \tau$ for all $N \in \mathcal{N}$.*

*There are three possible kinds of events:*

1. *In a* synchronous event *the system net transition $\widehat{t} \neq \widehat{\epsilon}$, fires synchronously with all the object net transitions $\vartheta(N), N \in \mathcal{N}$. At least one $N \in \mathcal{N}$ must exist with $\widehat{l}(\widehat{t})(N) \neq \tau$ and $\vartheta(N) \neq \epsilon$. We demand $\vartheta(N) \neq \epsilon \Leftrightarrow \widehat{l}(\widehat{t})(N) \neq \tau$ and that the channels have to match, i.e. $\widehat{l}(\widehat{t})(N) = l_N(\vartheta(N))$ for all $N \in \mathcal{N}$.*

2. *In a* system-autonomous event *$\widehat{t} \neq \widehat{\epsilon}$ fires autonomously. We demand that $\widehat{l}(\widehat{t})(N) = \tau$ for all $N \in \mathcal{N}$ and $\vartheta(N) = \epsilon$ for all $N \in \mathcal{N}$, i.e. $\vartheta = \vartheta_\epsilon$*

3. *In an* object-autonomous event *we demand $\vartheta(N) \neq \epsilon$ for exactly one object net $N$. Moreover, the transition $\vartheta(N)$ must not use a channel, that is $l_N(\vartheta(N)) = \tau$ has to hold.*

*The* set of synchronous events *is denoted by $\Theta_l$, the* set of system-autonomous events *is denoted by $\Theta_s$, and the* set of object-autonomous events *is denoted by $\Theta_o$. The* set of (system) events *is the disjoint union of these three sets:*

$$\Theta := \Theta_l \cup \Theta_s \cup \Theta_o$$

Note that for object nets which do not participate in a synchronous event (either because they are not in the preset of the system net transition or because no object net transitions fires synchronously) $\widehat{l}(\widehat{t})(N) = \tau$ holds, which forces $\vartheta(N) = \epsilon$ and thus $l_N(\vartheta(N)) = l_N(\epsilon) = \tau = \widehat{l}(\widehat{t})(N)$.

The requirements for a system-autonomous event imply $\vartheta(N) \neq \epsilon \Leftrightarrow \widehat{l}(\widehat{t})(N) \neq \tau$, the equivalence we had to demand in the case of a synchronous event. Moreover, $l_N(\vartheta(N)) = \widehat{l}(\widehat{t})(N)$ follows, too.

Also in an object-autonomous event the labels match again for all $N$, i.e. $\widehat{l}(\widehat{t})(N) = \widehat{l}(\widehat{\epsilon})(N) = \tau = l_N(\vartheta(N))$ for all $N \in \mathcal{N}$, but the equivalence $\vartheta(N) \neq \epsilon \Leftrightarrow \widehat{l}(\widehat{t})(N) \neq \tau$ does not hold for exactly one $N$, namely for the $N$ for which $\vartheta(N) \neq \epsilon$ holds. $\vartheta(N) \in T_N$ is the transition intended to fire object-autonomously.

If we write $\widehat{t}[\vartheta] \in \Theta$ in the following, this includes the possibility that the event is a system- or an object-autonomous event, i.e. $\vartheta = \vartheta_\epsilon$ or $\widehat{t} = \widehat{\epsilon}$ is possible. Moreover, since the sets of transitions are all disjoint, we usually write $\widehat{t}[\vartheta(N_1), \vartheta(N_2), \ldots]$ and also skip the object nets which are mapped to $\epsilon$, that is, we simply list the object net's transitions with which a system net transition synchronises.

**Example 3.5.** *Figure 3.3 below shows an* Eos *consisting of a system net* $\widehat{N}$ *and two object nets* $\mathcal{N} = \{N, N'\}$. *The typing of the system net is given by* $d(\widehat{p}_1) = d(\widehat{p}_2) = d(\widehat{p}_4) = N$ *and* $d(\widehat{p}_3) = d(\widehat{p}_5) = d(\widehat{p}_6) = N'$.

*For now, ignore the net-tokens on* $\widehat{p}_4, \widehat{p}_5$, *and* $\widehat{p}_6$. *These places are initially empty and the system has thus four net-tokens: two on place* $\widehat{p}_1$ *and one on* $\widehat{p}_2$ *and* $\widehat{p}_3$ *each. The net-tokens on* $\widehat{p}_1$ *and* $\widehat{p}_2$ *share the same structure, but have independent markings. The initial marking is thus given by*

$$\mu = \widehat{p}_1[\mathbf{0}] + \widehat{p}_1[a + b] + \widehat{p}_2[a] + \widehat{p}_3[a' + b'].$$

*We have two channels* $ch$ *and* $ch'$. *The labelling function* $\widehat{l}$ *of the system net is defined by* $\widehat{l}(\widehat{t})(N) = ch$ *and* $\widehat{l}(\widehat{t})(N') = ch'$. *The object net's labellings are defined by* $l_N(t) = ch$ *and* $l_{N'}(t') = ch'$. *Thus there is only one (synchronous) event:* $\Theta = \Theta_l = \{\widehat{t}[N \mapsto t, N' \mapsto t']\}$. *The event is also written shortly as* $\widehat{t}[t, t']$.

Projections are needed for e.g. ignoring the inner markings of net tokens in case of a system-autonomous event.

**Definition 3.6 (Projections).** *Let* $\mu \in \mathcal{M}_{OS}$ *be a nested marking of an* Eos *OS*. $\Pi^1(\mu)$ *denotes the projection of the nested marking* $\mu$ *to the system net level and* $\Pi^2_N(\mu)$ *denotes the projection to the marking belonging to the object*

*net $N$, i.e.*

$$\begin{aligned}\Pi^1(\sum_k \widehat{p}_k[M_k]) &= \sum_k \widehat{p}_k \text{ and}\\ \Pi^2_N(\sum_k \widehat{p}_k[M_k]) &= \sum_k \mathbf{1}_N(\widehat{p}_k)\cdot M_k\end{aligned}$$

*where $\mathbf{1}_N : \widehat{P} \to \{0,1\}$ with $\mathbf{1}_N(\widehat{p}) = 1$ iff $d(\widehat{p}) = N$.*

$\Pi^1(\mu)$ is again a *multiset*, i.e. a function $\widehat{P} \to \mathbb{N}$ such that $\Pi^1(\mu)(\widehat{p})$ is the number of (net) tokens that reside on $\widehat{p}$, but can also be seen as a vector from $\mathbb{N}^{|\widehat{P}|}$ (see Section 2.1).

Similarly $\Pi^2_N$ is also a multiset, it is the marking of the net $N$ viewed as a p/t net. Note, however, that $\Pi^2_N$ adds up all markings of object nets of type $N$ regardless to where these object net reside in the system net. $\Pi^2_N$ can also be seen as a vector from $\mathbb{N}^{|P_N|}$.

To explain firing we distinguish two cases: Firing a system-autonomous or synchronous event $\widehat{t}[\vartheta] \in \Theta_l \cup \Theta_s$ removes net-tokens together with their individual internal markings. The new net-tokens are placed according to the system net transition and the new internal markings are determined by the internal markings just removed and $\vartheta$. Thus a nested multiset $\lambda \in \mathcal{M}$ that is part of the current marking $\mu$, i.e. $\lambda \preceq \mu$, is replaced by a nested multiset $\rho$. The marking $\mu$ is not needed to define the enabling predicate, but is needed in the firing rule below.

**Definition 3.7** (**Enabling predicate**). *Let OS be an* Eos *and $\lambda, \rho \in \mathcal{M}$ be markings. Let $\widehat{t}[\vartheta] \in \Theta$ be an event. The enabling condition is expressed by the* enabling predicate *$\phi_{OS}$ (or just $\phi$ whenever OS is clear from the context). We distinguish two cases:*

1. *$\widehat{t}[\vartheta] \in \Theta_l \cup \Theta_s$ is a synchronous or system-autonomous event. In this case we have $\phi(\widehat{t}[\vartheta], \lambda, \rho)$ if and only if*

$$\begin{aligned}&\Pi^1(\lambda) = \mathbf{pre}(\widehat{t}) \wedge \Pi^1(\rho) = \mathbf{post}(\widehat{t})\\ \wedge\ &\forall N \in \mathcal{N} : \Pi^2_N(\lambda) \geq \mathbf{pre}_N(\vartheta(N))\\ \wedge\ &\forall N \in \mathcal{N} : \Pi^2_N(\rho) = \Pi^2_N(\lambda) - \mathbf{pre}_N(\vartheta(N)) + \mathbf{post}_N(\vartheta(N))\end{aligned}$$

   *where $\mathbf{pre}_N(\epsilon) = \mathbf{post}_N(\epsilon) = \mathbf{0}$ for all $N \in \mathcal{N}$.*

2. *$\widehat{\epsilon}[\vartheta] \in \Theta_o$ is an object-autonomous event. In this case let $N$ be the object net for which $\vartheta(N) \neq \epsilon$ holds. Now $\phi(\widehat{\epsilon}[\vartheta], \lambda, \rho)$ holds iff $\Pi^1(\lambda) = \Pi^1(\rho) = \widehat{p}$ for a $\widehat{p} \in \widehat{P}$ with $d(\widehat{p}) = N$ and $\Pi^2_N(\lambda) \geq \mathbf{pre}_N(\vartheta(N))$ and $\Pi^2_N(\rho) = \Pi^2_N(\lambda) - \mathbf{pre}_N(\vartheta(N)) + \mathbf{post}_N(\vartheta(N))$.*

In case of an object-autonomous event $\lambda$ and $\rho$ are thus essentially markings of an object net, but "preceded" by a system net place typed with this object net. Note that, in general, the event alone does not fully characterize the firing. For example, if an object net transition $t$ fires autonomously, the mode $\lambda$ is necessary, to describe where the object net resides. This is especially important, if two object nets of the same type exist on different system net places.

We are now ready to define the firing rule.

**Definition 3.8 (Firing Rule).** *Let $OS$ be an* Eos *and $\mu, \mu' \in \mathcal{M}$ markings. The event $\widehat{t}[\vartheta] \in \Theta$ is* enabled in $\mu$ for the mode $(\lambda, \rho) \in \mathcal{M}^2$*, denoted by $\mu \xrightarrow{\widehat{t}[\vartheta](\lambda,\rho)}$ iff $\lambda \preceq \mu \wedge \phi(\widehat{t}[\vartheta], \lambda, \rho)$ holds.*

*An event $\widehat{t}[\vartheta]$ that is enabled in $\mu$ for the mode $(\lambda, \rho)$ can fire: $\mu \xrightarrow[OS]{\widehat{t}[\vartheta](\lambda,\rho)} \mu'$. The resulting successor marking is defined as $\mu' = \mu - \lambda + \rho$.*

*As usual firing is extended to sequences $w \in (\Theta \cdot \mathcal{M}^2)^*$ inductively on the length of $w$*

- $\mu \xrightarrow{\epsilon} \mu$
- *If $w = w' \cdot \theta$ with $w' \in (\Theta \cdot \mathcal{M}^2)^*$ and $\theta \in (\Theta \cdot \mathcal{M}^2)$, then $\mu \xrightarrow{w} \mu'$ iff a marking $\mu''$ exists such that $\mu \xrightarrow{w'} \mu''$ and $\mu'' \xrightarrow{\theta} \mu'$ holds.*

*To denote that $\mu'$ is reachable from $\mu$ by some sequence of transitions we write $\mu \xrightarrow{*} \mu'$.*

*The set of reachable markings from a marking $\mu$ is denoted by $R(OS, \mu)$ or $R(\mu)$ is $OS$ is clear from the context. The set of reachable markings of $OS$, denoted by $R(OS)$, is the set of markings reachable from the initial marking $\mu_0$, i.e. $R(OS) := R(OS, \mu_0)$.*

*The reachability graph $RG(OS)$ is obtained as before for p/t net sytems, i.e. $RG(OS)$ is a directed graph where the set of nodes is the set of reachable markings and the (labelled) edges are the tuples $(\mu, \theta, \mu') \in \mathcal{M} \times \Theta \times \mathcal{M}$ where $\mu \xrightarrow{\theta} \mu'$.*

*We omit the mode and the* Eos *in the notations above if they are not relevant or clear from the context. We also say that $\widehat{t}[\vartheta]$ is* enabled in $\mu$ *or simply* active*, if a mode $(\lambda, \rho)$ exists such that $\widehat{t}[\vartheta]$ is enabled in $\mu$ for $(\lambda, \rho)$. This again is extended to sequences as above.*

**Example 3.9.** *To illustrate the firing rule, we return to the example of Figure 3.3. Note that the current marking $\mu$ enables $\widehat{t}[t, t']$ in the mode $(\lambda, \rho)$, where*

$$\begin{aligned} \mu &= \widehat{p}_1[\mathbf{0}] + \widehat{p}_1[a+b] + \widehat{p}_2[a] + \widehat{p}_3[a'+b'] = \widehat{p}_1[\mathbf{0}] + \lambda \\ \lambda &= \widehat{p}_1[a+b] + \widehat{p}_2[a] + \widehat{p}_3[a'+b'] \\ \rho &= \widehat{p}_4[a+2\cdot b] + \widehat{p}_5[\mathbf{0}] + \widehat{p}_6[c'] \end{aligned}$$

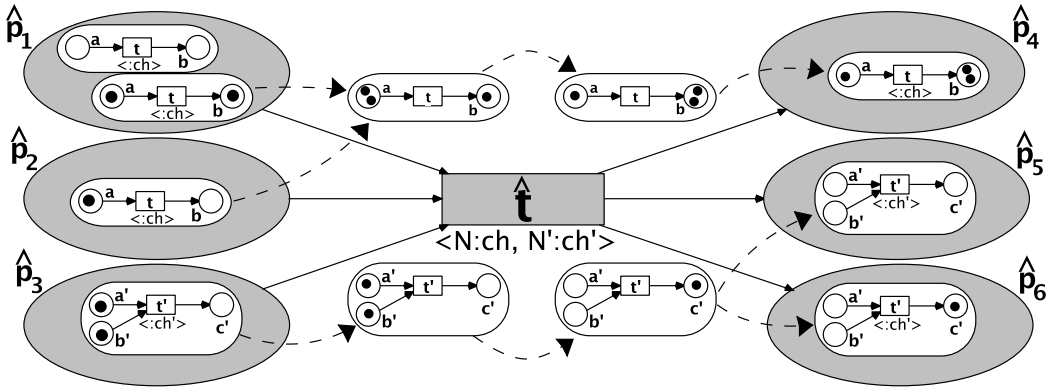


Figure 3.3: An Eos firing the synchronous event $\widehat{t}[t, t']$.

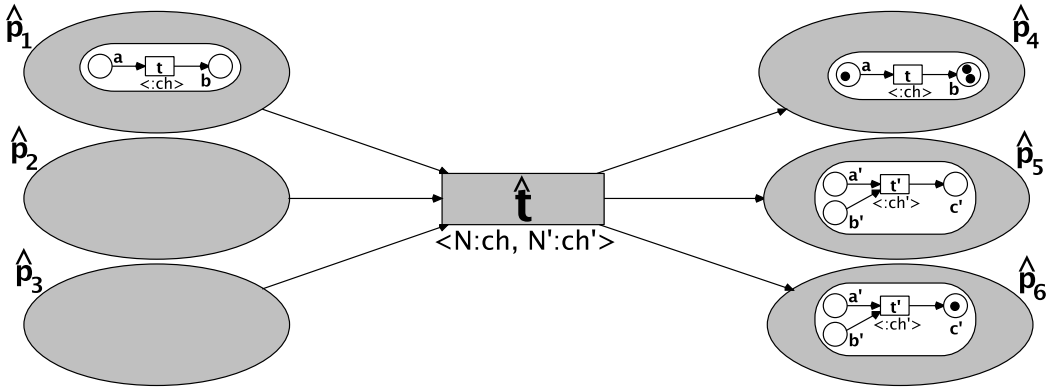


Figure 3.4: The Eos of Figure 3.3 after firing the synchronous event $\widehat{t}[t, t']$.

*The net-tokens' markings are added by the projections* $\Pi^2_N$ *and* $\Pi^2_{N'}$ *resulting in the markings* $\Pi^2_N(\lambda)$ *and* $\Pi^2_{N'}(\lambda)$*. Firing the object net's transitions generates the (sub-)markings* $\Pi^2_N(\rho)$ *and* $\Pi^2_{N'}(\rho)$*. This is illustrated above and below transition* $\widehat{t}$ *in Figure 3.3, where the left net on top is* $\Pi^2_N(\lambda)$ *and the right net on top is* $\Pi^2_N(\rho)$*. Similar for the nets below* $\widehat{t}$ *for the object net* $N'$*. After the synchronisation we obtain the successor marking* $\mu'$ *with new net-tokens on* $\widehat{p}_4$*,* $\widehat{p}_5$*, and* $\widehat{p}_6$*:*

$$\begin{aligned} \mu' &= (\mu - \lambda) + \rho = \widehat{p}_1[\mathbf{0}] + \rho \\ &= \widehat{p}_1[\mathbf{0}] + \widehat{p}_4[a + 2 \cdot b] + \widehat{p}_5[\mathbf{0}] + \widehat{p}_6[c'] \end{aligned}$$

*The result is shown in Figure 3.4.*

The firing rule uses a so called "distributed token semantics" in which the tokens of an object net may be distributed if copies of that object net are created during firing. Other semantics are possible, in particular, a value semantic where exact copies of an object net, including its internal marking, on an

input place of a system net transition are assigned to the output places of that transition (cf. [Val04] and Section 3.4 below).

The firing rule in both semantics incorporates the possibility to test if a net token's marking is the empty marking: If an object net of type $N$ appears in the preset of a system net transition $\widehat{t}$ once, and not in its postset, then $\widehat{t}$ can only fire if the net token is marked with the empty marking or is emptied by a synchronous event. Otherwise the equation $\Pi^2_N(\rho) = \Pi^2_N(\lambda) - \mathbf{pre}_N(\vartheta(N)) + \mathbf{post}_N(\vartheta(N))$ does not hold, since $\Pi^2_N(\rho) = \mathbf{0}$ due to $\Pi^1(\rho) = \mathbf{post}(\widehat{t})$ and there are no places of type $N$ in the postset of $\widehat{t}$, but $\Pi^2_N(\lambda) - \mathbf{pre}_N(\vartheta(N)) + \mathbf{post}_N(\vartheta(N)) \neq \mathbf{0}$. This feature is used in the proofs of Turing-completeness below and both semantics thus give rise to Turing-complete formalisms. Moreover, since the value semantics is easier from a computational point of view, because the inner markings of net tokens do not need to be distributed, the positive results in this work, especially the results in Chapter 5 hold for both semantics, too. From now on we will stick to the distributed token semantics as defined above.

In the definition of an Eos (Definiton 3.1) the labelling $l = (\widehat{l}, (l_N)_{N \in \mathcal{N}})$ is present. This is different from former definitions or the definitions in [KB11] which focuses on decidability issues and more accurate from a modelling point of view. Note that the space needed to store the labelling is small: With $T_s := \max\{|T_{N_1}|, \ldots, |T_{N_n}|, |\widehat{T}|\}$ the labelling of the object nets can be stored in an array of size $O(T_s \cdot |\mathcal{N}|)$ (each transition of each net has one label) the labelling of the system net can be stored in an array of size $O(T_s \cdot |\mathcal{N}|)$, too (each transition has at most one label for each object net), and thus the whole labelling can be stored in an array of size $O(T_s \cdot |\mathcal{N}|)$, which is polynomial in the other elements of an Eos. The number of generated events on the other hand, i.e. the size of the set $\Theta$, may become huge. To see this let $T_i$ be the set of transitions of the object net $N_i$. Let $\widehat{l}(\widehat{t})(N_i) = c_i$ for all $i$ and one system net transition $\widehat{t}$, where the $c_i$ are channels. Let $l_{N_i}(t) = c_i$ for all $t \in T_i$ and all $i$. Now $\widehat{t}$ may fire synchronously with each transition in $N_1$, each in $N_2$ and so on. Each of these possibilities results in a different event, so we already have at least $|T_1| \cdot |T_2| \cdot \ldots \cdot |T_n|$ events, a number exponential in the number of object nets and thus in the size of the Eos. Note that this is possible for each system net transition resulting in an even larger number of events.

While it is possible to construct an Eos with the above number of events, this number is also an upper bound, because any different labelling would reduce the number of possible object net transitions that may fire synchronous with an system net transition and thus would reduce the number of events (see also Lemma 4.10).

**Lemma 3.10.** *Let $OS = (\widehat{N}, \mathcal{N}, d, l)$ be an* Eos*. Let $|T| := \max\{|T_N| \mid N \in \mathcal{N}\}$. Then $|\Theta|$, i.e. the number of events, is bounded above by $|\widehat{T}| \cdot (|T| + 1)^{|\mathcal{N}|}$.*

*Proof.* An event is composed of a system net transition $\widehat{t} \in \widehat{T}$ and a mapping $\vartheta : \mathcal{N} \to T_N \cup \{\epsilon\}$ with $\vartheta(N) \in T_N$ if $\vartheta(N) \neq \epsilon$ for all $N \in \mathcal{N}$. There are thus at most $(|T|+1)^{|\mathcal{N}|}$ different mappings and thus $|\widehat{T}| \cdot (|T|+1)^{|\mathcal{N}|}$ possible events. □

Although the number of events is thus exponential in $|\mathcal{N}|$, it is easy to check with a couple of table lookups, if a given input $\widehat{\tau}[\vartheta]$ is indeed an event, that is if $\widehat{\tau}[\vartheta] \in \Theta$ holds. It is thus possible to enumerate all elements of $\Theta$ and use this to compute, given a marking $\mu$ of an Eos, all successor markings of $\mu$.

**Lemma 3.11.** *Given an* Eos *OS and a marking $\mu$ of OS, it is possible to compute all immediate successors of $\mu$.*

*Proof.* We enumerate all events and for each event $\widehat{\tau}[\vartheta] \in \Theta$ we enumerate all modes $(\lambda, \rho)$ such that $\lambda \preceq \mu$ holds and $\rho$ is in accordance with the enabling predicate (see Definition 3.7), that is $\rho$ satisfies $\Pi^1(\rho) = \mathbf{post}(\widehat{\tau})$ and $\Pi^2_N(\rho) = \Pi^2_N(\lambda) - \mathbf{pre}_N(\vartheta(N)) + \mathbf{post}_N(\vartheta(N))$ for all $N \in \mathcal{N}$. Note that for one $\lambda$ several modes $(\lambda, \rho)$ may exist, that is the markings of the object nets may be distributed in different ways in the successor marking of $\mu$. For each so chosen event $\widehat{\tau}[\vartheta]$ and mode $(\lambda, \rho)$ we check if $\widehat{\tau}[\vartheta]$ is indeed enabled in $\mu$ for this mode and, if so, compute the successor marking. □

Note that Lemma 3.11 also implies that it is possible to compute the reachability graph.

Moreover, by enumeration of events or modes it is also possible to compute all possible successor markings given a marking and an event, to test if a given event is active in a given marking, to test if a marking is a deadlock, and given two markings $\mu$ and $\mu'$ to check if an event exists that is active in $\mu$ and whose firing results in $\mu'$.

Also note that the algorithm in the proof of Lemma 3.11 is not very efficient. Instead of enumerating the events and modes it is also possible to firstly choose a system net transition whose input places are marked with net tokens. Then the event is constructed in accordance with the labelling of this system net transition. The modes are then not enumerate but chosen in accordance with the enabling predicate. Since more than one event and more than one mode might be possible, here, too, enumeration might be necessary, but to a far smaller degree than in the proof of Lemma 3.11. Moreover, due to the different distributions of the object nets markings that are usually possible due to the modes $(\lambda, \rho)$ present in the firing rule, one in general has to deal with exponential many successor markings in the size of the Eos even if the event is fixed.

We now define standard problems for Eos analog to problems for Petri nets.

**Definition 3.12 (Problems for Eos).** *1. In the* reachability problem for Eos *an* Eos *OS $= (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ and a marking $\mu$ of OS are given and the question is if $\mu_0 \xrightarrow{*} \mu$ holds.*

2. *In the* liveness problem for Eos *an* Eos $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *is given and the question is if the* Eos *is* live*, i.e. if all events $\theta \in \Theta$ are live. An event $\theta$ is live iff for all markings $\mu$ reachable from $\mu_0$ there exists a marking $\mu'$ reachable from $\mu$ that enables $\theta$.*

3. *In the* group liveness problem for Eos *an* Eos $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *and a system net transition $\widehat{t} \in \widehat{T}$ are given and the question is if $\widehat{t}$ is* group-live*, i.e. if for all markings $\mu$ reachable from $\mu_0$ there exists a marking $\mu'$ reachable from $\mu$ that enables $\widehat{t}[\vartheta]$ for some $\vartheta$.*

4. *In the* coverability problem for Eos *an* Eos $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *and a marking $\mu$ of $OS$ are given and the question is if a marking $\mu'$ exists such that $\mu' \in RS_{OS}(\mu_0)$ and $\mu' \geq \mu$ hold.*

5. *In the* boundedness problem for Eos *an* Eos $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *is given and the question is if $OS$ is* bounded*, i.e. if the set of reachable markings $RS_{OS}(\mu_0)$ is finite.*

For general Eos as defined in Definition 3.1 these problems turn out to be undecidable due to Turing-completeness of the formalism. This is the subject of the next section.

## 3.3 Turing-completeness of Eos

As illustrated in Section 3.1, Eos are quite helpful to model a variety of problems which exhibit a certain nesting of structures or a mobility of objects. Unfortunately, the formalism is Turing-complete, which can be shown by slightly modifying a proof from [Köh07]. There it is shown that for Eos with the additional requirement that each net token type that appears in the preset of a system net transition also appears in the postset of that transition, the reachability problem is undecidable (see also the treatment of conservative Eos in Section 4.1). This proof can be adapted to show that Eos are Turing-complete. Consequently, verification techniques are in general not applicable to unrestricted Eos.

We will give two proofs for Turing-completeness of Eos in this section. The first proof is via a direct simulation of counter machines resp. counter programs. The proof idea is from [Köh07]. We follow the presentation from [KB11].

The construction used in this proof will be used later on to prove variants of Eos to also be Turing-complete (cf. the discussion of unary and persistent Eos in Chapter 5).

The second proof is by a simulation of inhibitor nets and originates from the desire to also prove undecidability of problems like reachability and liveness in weaker formalisms like conservative Eos (see Chapter 4.1). This second proof is joint work with Michael Köhler-Bußmeier and has been published in [KBH12].

In both proofs the main idea is to use the null-test "hidden" in the firing rule: An object net $N$ may only be "removed" or "deleted" if its marking $m$ is the empty marking, i.e. if $m = \mathbf{0}$.

## Simulation of Counter machines by Eos

Counter machines and counter programs have been introduced in Chapter 2 (see Definitions 2.2 and 2.3). A counter program has access to a finite number of counters. Each counter can be increased or decreased by 1, where the last operation can only be applied if the counter's value is above 0. A third operation is a *jump* operation: if the counter's value is 0 the program jumps to a certain position in the program, otherwise to another. The last operation is the *halt* operation which terminates the program.

In the following we prove that counter programs can be simulated by Eos. The proof idea is from [Köh07]. We follow the presentation from [KB11].

**Theorem 3.13.** *Each counter program can be bisimulated by an* Eos.

*Proof.* We sketch how the four operations of a counter program can be simulated by an Eos. More details can be found in [KB11].

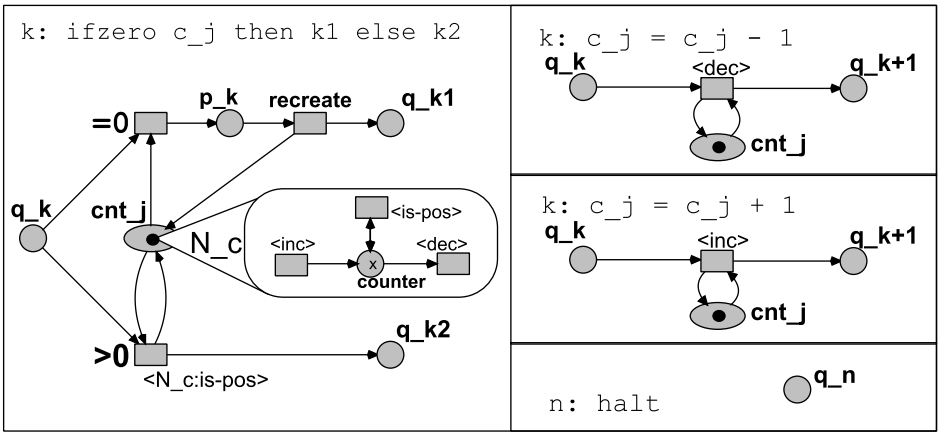


Figure 3.5: The Eos-translation of counter commands

The components of an Eos that simulate the four operations *increase*, *decrease*, *jump if zero*, and *halt* of a counter program are depicted in Figure 3.5. The places $q_i$ mark the current position in the counter program. The places $p_i$ are "intermediate places" between the places $q_k$ and $q_{k+1}$. Of all the $q_i$ and $p_j$ places exactly one is marked with a black token at any reachable marking. The places $cnt_j$ are marked with an object net each, which models the corresponding counter. The object net has one place *counter* which has $x$ black tokens.

For the increase operation a transition in the system net synchronises via the channel *inc* with a transition in the object net and a black token is added to the object net place *counter*. The decrease operation is modelled similarly.

The halt operation is simply modelled by a place $q_n$. If $q_n$ is marked by a black token, no further events are enabled.

Most interesting is the jump operation. The transition `=0` can only fire, if the object net on place $cnt_j$ is empty (otherwise the firing rule forbids that the net token is erased!). If the object net is empty, `=0` may fire and the net token for the counter is recreated afterwards. If the object net's place *counter* is marked, the transition `>0` synchronises with the object net via the channel *is-pos* and the synchronous event can fire. □

Basically the construction uses a null test "hidden" in the firing rule. In the postset of the transition `=0` no place typed with the same net as the place $cnt_j$ in its preset exists and so `=0` can only fire if the net token on $cnt_j$ is not marked, i.e. exactly zero black tokens are placed on the places of that net token.

If one demands that an object net type that appears in the preset of a system net transition also appears in its postset, the above construction is not possible anymore and indeed important problems become decidable. Unfortunately, reachability and liveness remain undecidable (cf. the discussion of *conservative* Eos in Chapter 4).

## Simulation of Inhibitor Nets by Eos

In the following we give an alternative proof of the Turing-completeness of Eos. We show that Eos are capable to simulate inhibitor nets (see the discussion of extensions to Petri nets in Section 2.4). The construction can then easily be adapted to also prove undecidability of the reachability and liveness problems of a variant, namely of conservative Eos. The following proof can be found in [KBH12] and originates from joint work with Michael Köhler-Bußmeier.

In the following we give a direct Eos-simulation of inhibitor nets which in turn can simulate counter programs as was proven by Hack [Hac76c].

**Theorem 3.14.** *For each inhibitor net $N^*$ there is an* Eos $OS_{strong}(N^*)$ *that bisimulates $N^*$.*

*Proof.* Without loss of generality we consider inhibitor nets without arc weights (see the discussion following this proof) and we assume that for each transition $t$ we have that whenever a place $p$ is connected with $t$ via an inhibitor arc then this place is not connected with $t$ via a normal arc. An inhibitor net is then given as $N^* = (P^*, T^*, F^*, F^*_{inh}, m_0)$, where $F^*_{inh} \subseteq P^* \times T^*$ describes the inhibitor arcs. The set of inhibitor places is $P^i := \{p^i \mid \exists t \in T^* : (p^i, t) \in F^*_{inh}\}$. A transition $t$ is enabled in $m$ iff there is at least one token on each input place

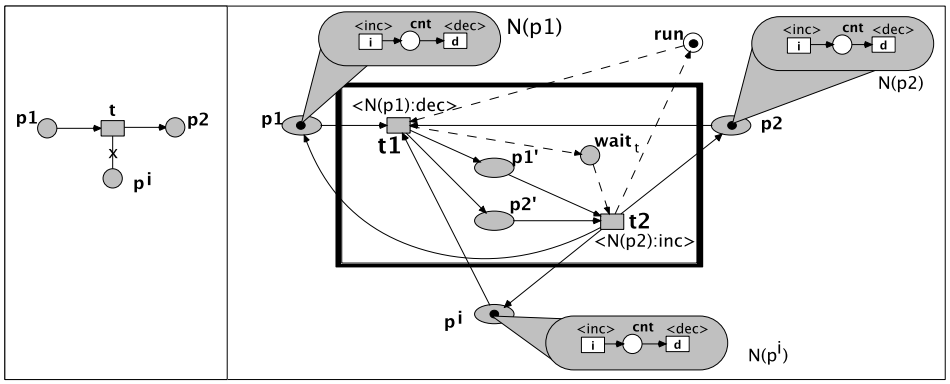


Figure 3.6: The Eos-Translation of Inhibitor Nets

and all inhibitor places are empty, i.e. $m(p) \geq F^*(p,t)$ for all $p$ and $m(p) = 0$ for all $p$ such that $(p,t) \in F^*_{inh}$.

The simulating Eos $OS_{strong}(N^*) = (\widehat{N}, \mathcal{N}, d, \Theta, \mu_0)$ is constructed in the following way (as illustrated in Figure 3.6):

- For each place $p \in P^*$ in the inhibitor net the simulating Eos has one object-net $N(p)$. Each object-net $N(p)$ has exactly one place $\mathsf{cnt}_{N(p)}$ and the two transitions $\mathsf{i}_{N(p)}$ and $\mathsf{d}_{N(p)}$, where $\mathsf{i}_{N(p)}$ is labelled with channel $\mathsf{inc}_{N(p)}$ and $\mathsf{d}_{N(p)}$ is labelled with channel $\mathsf{dec}_{N(p)}$. Note that all the object nets $N(p)$ have the same net structure. Additionally we have the object-net $\bullet$:

$$\mathcal{N} = \{\bullet\} \cup \{N(p) \mid p \in P^*\}$$

- The system net $\widehat{N}$ is obtained from the inhibitor $N^*$ via a substitution for each transition which is illustrated in Figure 3.6:

  Each transition $t \in T^*$ is replaced by two transitions $t_1$ and $t_2$.

$$\widehat{T} := \{t_1, t_2 \mid t \in T^*\}$$

  For each place $p \in P^*$ we add a copy $p'$. Additionally, we have one global *run* place and a set of *wait* places. These guarantee that firing of $t_1$ must be followed by $t_2$ before any other transition can fire.

$$\widehat{P} := P^* \cup \{p' \mid p \in P^*\} \cup \{\mathsf{run}\} \cup \{\mathsf{wait}_t \mid t \in T^*\}$$

  For each place $p$ connected with $t$ we have the arcs $(p, t_1)$ and $(t_2, p)$. For each place $p$ connected with $t$ with a normal arc we have the arcs $(t_1, p')$ and $(p', t_2)$. Furthermore, the run- and the wait place is connected with $t_1$ and $t_2$: $(\mathsf{run}, t_1)$, $(t_1, \mathsf{wait}_t)$, $(\mathsf{wait}_t, t_2)$, and $(t_2, \mathsf{run})$.

- The set of events $\Theta$ is generated from the labelling. It consists of all the events $t_1[\vartheta_1]$ with

$$\forall N(p) : \vartheta_1(N(p)) = \begin{cases} \mathsf{d}_{N(p)} & \text{if } (p,t) \in F^* \cap (P^* \times T^*) \\ \epsilon & \text{otherwise} \end{cases}$$

  and all the events $t_2[\vartheta_2]$ with

$$\forall N(p') : \vartheta_2(N(p')) = \begin{cases} \mathsf{i}_{N(p')} & \text{if } (t,p') \in F^* \cap (T^* \times P^*) \\ \epsilon & \text{otherwise} \end{cases}$$

  Note that for a given $t \in T^*$ the mappings $\vartheta_1$ and $\vartheta_2$ are uniquely defined.

- The typing $d$ is defined as $d(p) = d(p') = N(p)$ for all $p \in P^*$ and $d(\mathsf{run}) = d(\mathsf{wait}_t) = \bullet$.

Each marking $m$ of the inhibitor net is encoded as the marking $\mu(m)$ of the Eos. We say that a nested marking $\mu$ encodes a marking $m$ of $N^*$ whenever $\mu$ contains exactly one net-token on each place $p \in P^*$ (and none on the other places) and the net-token on $p$ has exactly $m(p)$ tokens on its place $\mathsf{cnt}_{N(p)}$:

$$\mu(m) := \mathsf{run}[] + \sum_{p \in P^*} p[m(p) \cdot \mathsf{cnt}_{N(p)}]$$

To finish the definition of $OS_{strong}(N^*)$ the initial marking encodes $m_0$, i.e. $\mu_0 := \mu(m_0)$.

Each firing $m \xrightarrow{t} m'$ is simulated deterministically by the firing of the events $t_1[\vartheta_1]$ and $t_2[\vartheta_2]$, i.e. by $\mu(m) \xrightarrow{t_1[\vartheta_1] \cdot t_2[\vartheta_2]} \mu(m')$. Whenever a place $p^i$ is connected via an inhibitor arc with $t$ then $t_1$ has exactly one place of type $N(p^i)$ in its preset but none in the postset. Therefore $t_1$ can only fire if the marking of the net-token is the empty multiset. Whenever $t_2$ fires it generates one net-token on $p^i$ again which must be empty, since there is no place of type $N(p^i)$ in the preset of $t_2$. It is straightforward to see that we have:

$$m \xrightarrow{t} m' \iff \mu(m) \xrightarrow{t_1[\vartheta_1] \cdot t_2[\vartheta_2]} \mu(m')$$

This proves that the Eos $OS_{strong}(N^*)$ simulates the inhibitor net $N^*$. □

The restriction at the beginning of the proof to only consider inhibitor nets without arc weights is not severe. The proof can be adjusted for the general case by adding additional transitions $\mathsf{i}^k_{N(p)}$ and $\mathsf{d}^k_{N(p)}$ for all $k \leq n$, where $n$ is the greatest arc weight occuring in $N^*$. The object net transition $\mathsf{i}^k_{N(p)}$ adds exactly $k$ tokens to the place $\mathsf{cnt}_{N(p)}$ while $\mathsf{d}^k_{N(p)}$ removes $k$ tokens from it to simulate the effect of arc weights.

We now turn to the liveness problem for Eos.

**Lemma 3.15.** *The reachability problem for inhibitor nets is reducible to the liveness problem for* Eos.

*Proof.* The proof follows the idea given in [Pet81] showing the equivalence of reachability and liveness for p/t nets.

It is sufficient to consider the problem whether the empty marking is reachable, since for each inhibitor net $N_1$ and each marking $m$ we can construct another inhibitor net $N_2$ with the property: The marking $m$ is reachable in $N_1$ iff $\mathbf{0}$ is reachable in $N_2$. The net $N_2$ is obtained from $N_1$ by adding one place $\mathsf{run}$ and one transition $t$. The additional $\mathsf{run}$-place is attached as a side condition to each transition of $N_1$. Initially the place $\mathsf{run}$ is marked with one token. The additional transition $t$ removes exactly $m(p)$ tokens from each $p$ (where $m$ is the given marking that is tested for reachability) and one token from $\mathsf{run}$. The postset of $t$ is empty. It is obvious that $N_2$ has the desired property.

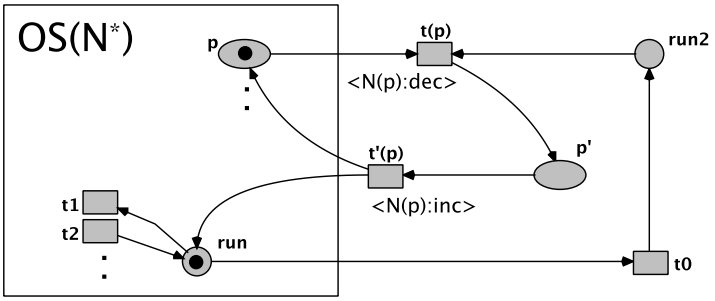


Figure 3.7: Reduction from $\mathbf{0}$-Reachability to Liveness of $t_0[\mathbf{0}]$

We will construct an Eos $OS(N^*)$ from a given inhibitor net $N^*$ such that the empty marking is reachable in $N^*$ iff the event $t_0[\mathbf{0}]$ is not live in $OS(N^*)$.

Assume the inhibitor net is given as $N^* = (P^*, T^*, F^*, F^*_{inh}, m_0)$. We define $OS(N^*)$ as an extension of $OS_{strong}(N^*)$ from Theorem 3.14 (see Fig. 3.7): We add the transition $t_0$ and the place $\mathsf{run}_2$ and for each $p \in P^*$ the place $p'$ and the transitions $t(p)$ and $t'(p)$ with the arcs $(p, t(p))$, $(t(p), p')$, $(p', t'(p))$, $(t'(p), p)$, $(\mathsf{run}_2, t(p))$, $(t'(p), \mathsf{run})$. Furthermore, we add the arcs $(\mathsf{run}, t_0)$ and $(t_0, \mathsf{run}_2)$. We set $d(\mathsf{run}_2) := \bullet$ and $d(p') := d(p) = N(p)$.

Note that since $t_0$ has only places of the black token type in the pre- and postset, we obtain that if the event $t_0[\vartheta]$ is activated, then $\vartheta = \mathbf{0}$.

As before, we define $\mu(m)$ as:

$$\mu(m) := \mathsf{run}[] + \sum_{p \in P^*} p[m(p) \cdot \mathsf{cnt}_{N(p)}]$$

A marking that is reachable in $N^*$ is so in $OS(N^*)$:

$$m_1 \xrightarrow{*} m_2 \Longrightarrow \mu(m_1) \xrightarrow{*} \mu(m_2)$$

Assume that $\mathbf{0}$ is reachable in $N^*$. In $\mu(\mathbf{0})$ we have $\mu(\mathbf{0}) \xrightarrow{t_0[\mathbf{0}]} \mu := \mathsf{run}_2[] + \sum_{p \in P^*} p[\mathbf{0}]$ and in $\mu$ no event is activated anymore. So, if $\mathbf{0}$ is reachable in $N^*$, then, clearly, $t_0[\mathbf{0}]$ is not live.

Assume that $\mathbf{0}$ is not reachable in $N^*$. Then $t_0[\mathbf{0}]$ is live: For each marking $m^* \neq \mathbf{0}$ we have $m^*(p) > 0$ for some $p$ and therefore we have $\mu(m^*) \xrightarrow{t_0[\mathbf{0}]} \mu' \xrightarrow{t(p)[\vartheta] \cdot t'(p)[\vartheta']} \mu(m^*)$, where $\vartheta, \vartheta'$ are clear from Figure 3.7. Note that the sequence $t(p)[\vartheta] \cdot t'(p)[\vartheta']$ does not alter the marking of the net-token on $p$. □

Putting the results of the above Lemma 3.15 and Theorem 3.14 together we have:

**Theorem 3.16.** *Reachability, liveness, group-liveness, coverability, and boundedness are undecidable for* Eos.

*Proof.* Undecidability of the reachability problem follows from Theorem 3.14 and the fact that reachability is undecidable for inhibitor nets.

Undecidability of the liveness problem then follows from Lemma 3.15. From the proof of Lemma 3.15 we can see that group-liveness is also undecidable, since the empty marking is reachable in $N^*$ iff $t_0$ is not group-live in $OS(N^*)$.

Undecidability of coverability follows similarly from the undecidability of the corresponding problem for inhibitor nets.

For boundedness, note that for each reachable marking $\mu(m)$ each place in $OS_{strong}(N^*)$ contains at most one net-token and all net-tokens are bounded iff the inhibitor net is bounded. Therefore, $OS_{strong}(N^*)$ is bounded iff $N^*$ is bounded. Since boundedness is undecidable for inhibitor nets, it is undecidable for Eos.

□

More things about Eos could be said. For example, the firing rule is not *monotone*, because an empty net token might be removed by a transition, but adding tokens to that net token and thus increasing the overall marking, the system net transition might now be unable to fire. Thus the usual construction of a coverability graph is not possible, which is little surprising, since coverability is undecidable for Eos.

Furthermore, one can show, that the firing rule is *reversible*, i.e. when all arcs are reversed one can fire "backwards", that Eos can be seen as a *canonical extension of p/t nets*, and that it is possible to establish an *invariance calculus for* Eos based on place invariants for p/t nets. For these topics see the report by Köhler-Bußmeier [KB11].

Semantics that differ from the distributed token semantics employed here are described in [Val04] and [KF07]. These are also discussed in Section 3.4 below.

While the above mentioned results are interesting, they are not needed in the following. The treatment of Eos in this chapter suffices for our purposes.

## 3.4 Related Approaches

As mentioned in the introduction several formalisms are known which try to capture the idea of mobility. The idea apparent in the formalism of Eos to use Petri nets as tokens originates from [Val91] and is extended to *elementary object systems* in [Val98]. The formalism in [Val98] is very close to the definition of Eos given here. Appart from distributed token semantics, as used in this thesis, *reference* and *value semantics* are also investigated. In value semantics a net token together with its internal marking is treated as a value. Without any distribution of the tokens of its internal marking, exact copies of an object net on an input place of a system net transition are assigned to the output places of that transition. In reference semantics a token in the system net is a reference to an object net. These references may then be copied, but they point to the same object net. Another semantic, *mobility semantic*, is introduced in [KF07]. A conversion equivalence allows the exchange of tokens between different net tokens prior to firing. See [Val04] and [KF07] for a detailed explanation.

*Unary* Eos are Eos with only one object net type. In [KR04] it is shown that the reachability problem for unary Eos is decidable, if a reference semantic is used, but undecidable, if a distributed token semantic is used.

In a formalism called *generalised state machines* (GSM), another restriction of Eos, each object net appears exactly once in the Eos and thus the above mentioned semantics are identical. GSMs are introduced in [KR05] (there called *ordinary object-net systems*). They are an interesting formalism, because many important problems become decidable for them, but they still retain the main idea that a token can have inner activity. They are treated in detail in Chapter 4.

In their original form and as defined in Section 3.2 elementary object systems have a two levelled structure. In [KR03a] and [KR04] they are generalised to have an *arbitrary nesting structure*. In [KR03a] a reference semantic is used, in [KR04] a distributed token semantic (there called value semantic) is used and Turing-completeness of the formalism is shown. In both publications the nesting is allowed to be arbitrary and in particular is not bounded. Such a restriction is investigated in Chapter 6.

Further enhancing the formalism, in [KBH09] and [HKB12a] *object net systems* with an arbitrary nesting as above are introduced which are additionally equipped with channels that allow to transfer net-tokens in the vertical dimension of the nested marking. See also Chapter 6.

While the formalisms mentioned above have been used for modelling, see e.g. [KMR03], [KR05], [Val04], [KR03b], complexity-theoretic investigations are limited to decidability results (apart from the publications mentioned above see also [Köh07] and [Köh04]).

Similar to Eos are *minimal object-based nets* (MOB nets) [Kum00]. Tokens there, however, only have an identity and no further structure. The minimal

assumptions that tokens have a unique identity, which can be compared, and that new tokens with a new identity can be created is enough to prove that reachability is undecidable. This result then also holds for any formalism with name creation. For example, *$\nu$-abstract Petri nets* [RVFE08] are quite similar to MOB nets in that they allow the creation of fresh names and here the reachability problem is again undecidable. For subclasses of MOBs the reachability problem can again be decided [DK06].

Another generalisation of Eos are *reference nets* [Kum02] which employ a reference semantic and allow arbitrary pointer structures.

*Extended elementary object systems* introduced in [MTM04] and [MT08] are another extension of Eos. Most notably, the nesting depth is unrestricted and multiple system nets may be present. Further additions to the formalism simplify modelling and are specific to the application domain, i.e. the modelling of security mechanisms. To analyse the model, it is translated into a coloured Petri net and there a simulation-based analysis is undertaken, cf. [MTM04], [MT08], [DTMZ08] and [DTMZ10] and also [MT06] and [Ma05] for a book and a dissertation on extended elementary object systems.

Also similar to Eos are $PN^2$ from [Hir02] which are basically Eos with a value semantics and the *nested nets* from Lomazova [Lom00]. Nested nets, however, are a combination of Eos and coloured Petri nets. A token of the system net may be an object net or, for example, an integer. Furthermore, variables may be used at the arcs. Nested nets can easily simulate reset nets, because a net token can be removed regardless of its marking, and thus reachability, among others, is undecidable. Research on nested nets currently goes into the direction of *compositionality*, i.e. what structure is allowed for the system net and the object nets such that the composed nested nets is, e.g., live, cf. [DL12].

Different from Eos in that they allow the exchange of object nets and for the same reason similar to the object net systems from [KBH09] and [HKB12a] are *Petri hypernets* introduced in [BBPP05]. Petri hypernets employ a value semantic and do not allow the creation or destruction of net tokens. In this regard they are similar to generalised state machines. Moreover, Petri hypernets enjoy a finite state space and thus reachability, liveness and so on, are all decidable. There has also work been done on the verification of hypernets: In [BJP06] a model checking procedure for CTL over Petri hypernets is devised that runs in polynomial space establishing a similar result for Petri hypernets as our result in Chapter 5 for safe Eos. Safe Eos, however, allow the creation and destruction of net tokens. It is unclear, if the above result still holds, if Petri hypernets are adjusted in this regard. In [BJP12] Petri hypernets are extended to allow the creation of unstructured tokens and while the reachability problem remains decidable, its complexity is inherited from p/t nets and thus ExpSpace-hard.

In [CH94] coloured Petri nets are also enhanced with channels. This, however, does not lead to a nested formalism in the sense above. Moreover, only

the transport of *information*, i.e. of a coloured token, is allowed and not the transport of a net token. This formalism, however, might be combined with the nested nets [Lom00] discussed above to yield a similar formalism to the object net system with channels from [KBH09].

*Multi Agent Nets* (MANs) [MK96] and *mobile predicate/transition nets* are two further extensions of coloured Petri nets. Both are again members of the nets-within-nets family and in both a reference semantic is used to model nesting. Moreover, in MANs interaction is modelled by passing of tokens and in mobile predicate/transition nets places represent logical predicates.

Further formalisms based on Petri nets are *AHO systems* [HM03], [HEM05], *Mobile Systems* [Lak05], and *recursive nets* [HP99]. AHO systems allow complex data types as tokens and it is possible to encode net tokens as data types. However, it seems that this only works for a two level structure. Mobile Systems are a generalisation of modular Petri nets. A net is divided into modules. These modules can be nested and thus provide a notion of locality. The modules and submodules may shift from one module to another and may interact via place and transition fusion. In recursive nets a net may generate itself as a new thread by special transitions. In this way, a nested structure of threads develops and the nested marking resembles those of a object net system. However, only one net type is present and the interaction between the different threads is limited to creation and destruction.

All formalisms mentioned so far are based on Petri nets. Another line of research is concerned with process calculi which attempt to capture mobility and mobile computation. The *$\pi$-calculus* [Mil99] is a prominent example of these process calculi. However, at the 2001 *Application and Theory of Petri Nets* conference Robin Milner gave a keynote [Mil01]. He stressed that, in his view, mobility is concerned with the interplay between locality and connectivity, and that locations should be arbitrarily nested. Milner observed in his talk that the $\pi$-calculus was inadequate because it did not respect this nesting of locations [KB12b].

Two other prominent calculi are the *Ambient Calculus* by Cardelli and Gordon [CG00b] and the *Seal Calculus* by Castagna, Vitek, and Nardelli [CVN05]. In the Ambient Calculus processes reside in a hierarchy of locations, called ambients. Ambients can move in and out of each other and to new locations and thus satisfy Milner's requirements above. Cardelli and Gordon also proposed an Ambient Logic [CG00a] to reason about time and space in their calculus. 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. In [CZG+03] it is proven that the model checking problem for the Ambient Calculus against the Ambient Logic is under certain conditions PSPACE-complete. In particular, the calculus is replication-free, i.e. the set of reachable processes is finite.

The Seal Calculus [CVN05] also aims at describing mobile computation. One of the major differences between the Seal Calculus and the Ambient Calculus is that mobility in the Ambient Calculus is *subjective*, i.e. an agent moves himself, while it is *objective* in the Seal Calculus, i.e. an agent is moved by the context in which it resides, cf. [CVN05].

By now, many more calculi have been established and many interesting results are known. A nice bibliography on mobile processes is [Zil01].

Turning again to Petri net style formalisms, *Ambient Petri Nets* [FEA03a], [FEA03b] are an extension of Eos which allow an arbitrary nesting, as well. The aim here, however, is to provide a denotational semantics for the Ambient Calculus and also for a new calculus, the *Ambient Petri Box Calculus*, also discussed in [FEA03a] and [FEA03b].

Finally, another extension to the formalisms described above is the notion of *adaptiveness*. Nested nets have been extended with transitions that allow to create new net tokens out of existing ones in [LvHO+06]. Operations are the parallel composition of two nets, a choice operator, and sequential composition. In [KB09] the object net systems from [KBH09], that allow a vertical transport of tokens, are extended in a similar fashion to allow modification of net tokens at run-time. The *extended nested nets* from [LvHO+06] as well as the *Hornets* from [KBH09] are illustrated by practical relevant examples. Hornets are used to model a distributed workflow management system, extended nested nets to model a health care workflow. Both formalisms are Turing-complete. Therefore, the search for subclasses that are accessible for verification is an important research question. Some first results are presented in [Lom08] concerning extended nested nets and in [KB12a] where safe, elementary Hornets are investigated.

In summary, apart from the PSpace results for Petri hypernets [BJP06] and the Ambient Calculus [CZG+03], most results for the above formalisms are decidability (or undecidability) results. Our results in Chapter 5 complement the above two results for object net systems and go beyond them in that the creation of net tokens is allowed in our setting.

## 3.5 Summary

In this chapter, we motivated and formally introduced *Elementary Object Systems*, a formalism of the nets-within-nets family that, as defined here, makes use of a value semantic with regard to the net tokens and whose nesting depth is limited to two levels, the system net and the object net level. Eos where first introduced in [Val91], while our definitions follow [KB11]. Different from the presentation there, our definition here exhibits the labelling $l = (\widehat{l}, (l_N)_{N \in \mathcal{N}})$ instead of the set of events $\Theta$ which is more natural from a modelling point of view and also more amenable to complexity theoretic analysis, because $\Theta$ tends

to be very big and thus statements about the complexity of some procedure *in the size of the object net system* would not be helpful if $\Theta$ would be part of the object net system.

In Section 3.4, we surveyed related approaches like nested nets and hypernets which employ a Petri net view and with related approaches like the Ambient Calculus or the Seal Calculus which employ a process algebraic view.

Most of these formalisms are Turing-complete. In this chapter, we proved that Eos are Turing-complete in two ways: Firstly, by simulation of counter programs, see Theorem 3.13. The proof idea was from [Köh07]. Secondly, by simulation of inhibitor nets, see Theorem 3.14. The new proof originates from joint work by Michael Köhler-Bußmeier and me and can be found in [KBH12]. The construction in this alternative proof is helpful in the context of conservative Eos (see Section 4.1).

From these results we deduce undecidability of the reachability, the liveness, the group-liveness, and the boundedness problem for Eos, see Theorem 3.16.

While elementary object systems are thus helpful, modelling applications which arise in computer science and other fields and which exhibit nesting of structures or mobility of objects, they are, unfortunately, Turing-complete and general analysis techniques are out of reach. It is therefore a reasonable goal to find restrictions of the formalism which still allow to model a variety of applications in a comfortable way, but which also result in a formalism such that the models can be automatically analysed and checked for certain properties using affordable resources. These restrictions and the tradeoff between modelling capability and fast verification algorithms are the topic of the next chapters.

# 4 Structural Restrictions of Eos

In this chapter, we introduce several structural restrictions of the Eos-formalism.

At first we introduce conservative Eos which do not allow the creation or destruction of net tokens anymore. Unlike Eos, conservative Eos are not Turing-complete anymore, but they still suffer from an undecidable reachability and liveness problem.

We then take the idea employed for conservative Eos one step further and introduce generalised state machines (GSMs) which additionally do not allow a "split" or "join" of different net tokens' markings anymore. For GSMs many problems, among them reachability, liveness, coverability, and boundedness, become decidable, but solving them requires impractical resources. For that reason we then discuss several structural restrictions of GSMs and discuss the reachability problem for each of them. We introduce GSMs where the system net and all object nets are P- or T-nets, GSMs where all nets are acyclic, conflict-free GSMs, and free-choice GSMs. These structural restrictions are known for p/t nets and are carried over to GSMs here. We also introduce new restrictions only sensible in the context of object net systems. Namely, we introduce deterministic and strongly deterministic GSMs, which restrict the usage of channels and thus the possible events.

It turns out that the ability to synchronise the system net with the object nets is a significant source of complexity. The reachability problem tends to be hard to solve even for heavily restricted formalisms. For example, the reachability problem for a GSM where the system net and all object nets are P-nets, is PSpace-complete. For standard P-nets this problem is easily solvable in polynomial time and in the setting described above there is actually only *one* object net. So the synchronisation between one system net, which is a P-net, and one object net, which is a P-net, is enough to render the problem PSpace-hard. This is even the case for strongly deterministic GSMs, for which the usage of channels is heavily restricted. If the system net is a free-choice net, which treated in isolation is live, bounded and cyclic and the object nets are T-nets, then the reachability problem turns out to be even ExpSpace-hard, even if for all these nets in isolation the reachability problem can again be solved in polynomial time. Thus structural restrictions alone seem insufficient to make the formalism suitable for analysis.

In Section 4.1, we present conservative Eos and show that boundedness and coverability become decidable, while liveness and reachability remain undecid-

able. While conservative Eos have been introduced before in [Köh07] (not explicitly named conservative there, but the condition is the same) and the reachability problem for them proven to be undecidable (also in [Köh07]), the constructions with inhibitor nets shown here is new as is the proof of undecidability of the liveness problem for conservative Eos. These new constructions and results are joint work with Michael Köhler-Bußmeier and published in [KBH12].

In Section 4.2, we present generalised state machines and describe the construction of the so called *reference net* that allows to decide all problems, which are also decidable for p/t nets. The results of this section have been developed by Köhler-Bußmeier in [KR05] and are only slightly adapted here.

In Section 4.3, deterministic and strongly deterministic GSMs are introduced and the reachability problem for several structural restrictions of them is investigate in the following Sections 4.4 to 4.7. The formalisms and results in this section are my own contribution and joint work with Michael Köhler-Bußmeier. The results concerning P- and T-nets in the context of GSMs have been published in [HKB12b] and [HKB11b]. The results concerning conflict-free GSMs have been published in [HKB11a]. There deterministic and strongly deterministic GSMs have also been introduced for the first time. The generalisation to deterministic and strongly deterministic Eos and the treatment of these formalisms and of deterministic and strongly deterministic conservative Eos in Section 4.3 as well as the results concerning acyclic GSMs and free-choice GSMs presented in Sections 4.5 and 4.7 are new and appear in this work for the first time.

The chapter ends with a summary in Section 4.8.

## 4.1 Conservative Eos

In the following, we introduce conservative Eos and show that coverability, boundedness, and termination become decidable for them. The formalism is thus not Turing-complete anymore. However, we show that reachability and liveness remain undecidable. While conservative Eos have been known before as has the result on reachability (cf. [Köh07]), the proofs presented here are new and are joint work with Michael Köhler-Bußmeier. They have been published in [KBH12].

In a conservative Eos each object net type that appears in the preset of a system net transition $\widehat{t}$ also has to appear in the postset of $\widehat{t}$. Note that in both proofs that establish Turing-completeness of Eos (see Theorems 3.13 and 3.14) it is crucial that a net-token can be removed, if its marking is the null marking $\mathbf{0}$: In Figure 3.5 the transition =0 can only fire if the object net on place $cnt_j$ is empty, i.e. the place *counter* is not marked of the corresponding object net. In the same way the transition $t_1$ in Figure 3.6 can only fire if the object net on

place $p^i$ is empty, which exactly correspond to the place $p^i$ in the *inhibitor net* being empty. This firing of transitions can be seen as a null test, because the system net transition can only fire if the marking of a net token that resides on one of its input places is **0**.

In a conservative Eos such constructions are not possible anymore and it turns out that this was the source of the equivalence to inhibitor nets and counter programs. For conservative Eos the firing rule will again have the monotonicity property and it will be possible again to construct coverability graphs for them (cf. [KB11]).

**Definition 4.1** (**Conservative Eos**). *A typing is called* conservative *iff for each place $\widehat{p}$ in the preset of a system net transition $\widehat{t}$ such that $d(\widehat{p}) \neq \bullet$ there is a place in the postset being of the same type: $(d(\widehat{{}^\bullet t}) \cup \{\bullet\}) \subseteq (d(\widehat{t^\bullet}) \cup \{\bullet\})$, i.e. each object net type that appears in the preset of $\widehat{t}$ also appears in its postset.*

*An* Eos *is* conservative *iff its typing $d$ is.*

Note that the usage of the term "conservative" is different from the usage in the context of p/t nets (see Definition 2.42). There the number of tokens (or in the general case: the *weighted* number of tokens) remains constant. Here we "conserve" an object net *type*.

Boundedness, coverability, and termination, i.e. the problem if a marking is reached in which no event is enabled, can be decided for conservative Eos, showing that the formalism can not be Turing-complete, since termination can not be decided for Turing machines.

**Theorem 4.2.** *Boundedness, coverability, and termination are decidable for conservative* Eos.

*Proof.* In [KB11] it is shown that the firing rule is monotonic again, i.e. if an event $\theta$ is activated in a marking $\mu$ it is also activated in greater markings $\mu' > \mu$. It is then shown [KB11] that the reachability graph of a conservative Eos is a well structured transition system.

Generalising the result of Karp and Miller [KM69] it is shown in [FS01] that the boundedness, the coverability, and the termination problem are decidable for well structured transition systems and thus they are decidable for conservative Eos, too. □

While not being Turing-complete due to Theorem 4.2 above, we show in the following that reachability and liveness remain undecidable for conservative Eos.

For reachability this result has already been shown in [Köh07], where a weak simulation of counter programs is given. But the simulation of inhibitor nets given here results in a representation that is more accessible to the liveness problem.

The simulation is a "weak one" in the sense that wrong guesses about the test on zero are possible, but all false guesses are stored in the marking till the end.

We can reuse the construction presented in Theorem 3.14.

**Lemma 4.3.** *For each inhibitor net $N^*$ there is a conservative* Eos $OS_c(N^*)$ *such that each marking $m$ of $N^*$ is encoded as the* Eos *marking $\tilde{\mu}(m)$ with the following property:*

$$m \xrightarrow[N^*]{*} m' \Longrightarrow \tilde{\mu}(m) \xrightarrow[OS_c(N^*)]{*} \tilde{\mu}(m')$$

*Moreover, for each firing $\tilde{\mu}(m) \xrightarrow{*} \mu$ such that $\mu$ does not correspond to any marking in the inhibitor net, no marking reachable from $\mu$ will ever do so.*

*Proof.* Let us consider an inhibitor net given as $N^* = (P^*, T^*, F^*, F^*_{inh}, m_0)$, where $F^*_{inh} \subseteq P^* \times T^*$ describes the inhibitor arcs.

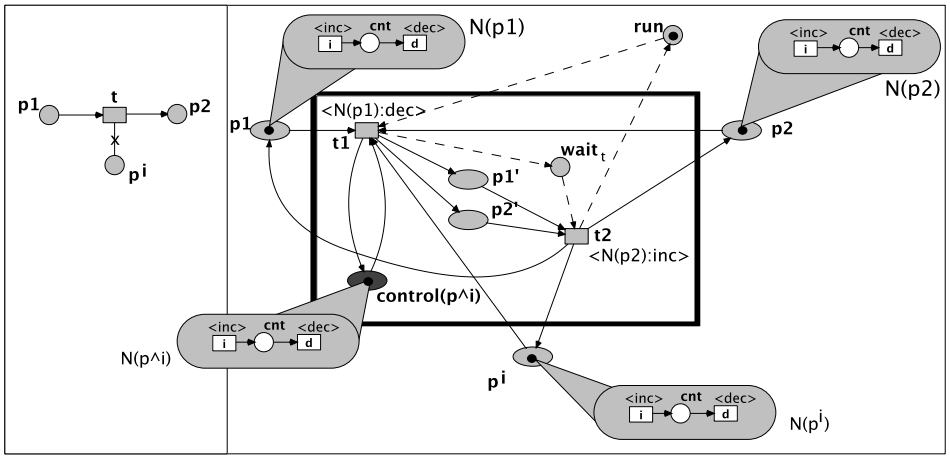


Figure 4.1: The Conservative Eos-Translation of Inhibitor Nets

The simulating Eos $OS_c(N^*)$ (cf. Figure 4.1) is obtained by minor modifications from the Eos $OS_{strong}(N^*)$ from Theorem 3.14 (cf. Figure 3.6): In addition to the places in $OS_{strong}(N^*)$ we add the system net places $\mathsf{control}(p^i)$ with $d(\mathsf{control}(p^i)) = N(p^i)$ for each inhibitor place $p^i$.

$$\widehat{P} := P^* \cup \{p' \mid p \in P^*\} \cup \{\mathsf{run}\} \cup \{\mathsf{wait}_t \mid t \in T^*\} \cup \{\mathsf{control}(p^i) \mid p^i \in P^i\}\}$$

For each inhibitor place $p^i \in P^i$ we add $\mathsf{control}(p^i)$ as a side condition to $t_1$.

The definition of the encoding of a marking $m$ of $N^*$ has to be adjusted, too: We say that a nested marking $\mu$ encodes $m$ whenever $\mu$ contains exactly one

net-token on each place $p \in P^*$, the net-token on $p$ has exactly $m(p)$ tokens on its place $\mathsf{cnt}_{N(p)}$, and one empty net-token on each control place:

$$\begin{aligned}\tilde{\mu}(m) &:= \mu(m) + \textstyle\sum_{p\in P^i} \mathsf{control}(p)[\mathbf{0}] \\ &= \mathsf{run}[] + \textstyle\sum_{p\in P^*} p[m(p)\cdot \mathsf{cnt}_{N(p)}] + \sum_{p\in P^i} \mathsf{control}(p)[\mathbf{0}]\end{aligned}$$

As before, the initial marking is defined as the encoding of $m$, i.e. $\mu_0 := \tilde{\mu}(m_0)$. By construction, the simulating Eos $OS_c(N^*)$ is conservative.

Assume that $m$ enables $t$ in the inhibitor net. Then the corresponding marking $\tilde{\mu}(m)$ enables $t_1[\vartheta_1]$ as before, since there are enough tokens in the preset. Since the run-place is emptied after the firing of $t_1[\vartheta_1]$, the simulation of other transitions is disabled. As before $t_2[\vartheta_2]$ generates the correct successor marking. For each inhibitor place $p^i$ the event $t_1[\vartheta_1]$ combines the net-token on $p^i$ with that from $\mathsf{control}(p^i)$. Since $t$ is activated in $N^*$, the place $p^i$ must be empty and therefore the net-token on $p^i$ in the Eos is empty, too. After firing $t_1[\vartheta_1]$ we have an empty net-token on each $\mathsf{control}(p^i)$ again. After that, $t_2[\vartheta_2]$ regenerates an *empty* net-token on each $p^i$. Additionally, $t_2[\vartheta_2]$ puts a token back on the run-place and the simulation can continue.

Conversely, it is not guaranteed that the inhibitor places are empty whenever $t_1[\vartheta_1]$ is enabled, i.e. for the test on zero we have to guess. What happens if some inhibitor place $p^i$ is not empty when $t_1[\vartheta_1]$ fires? In this case $t_1[\vartheta_1]$ produces a marking $\mu_1$ which has a non-empty net-token on $\mathsf{control}(p^i)$. After that, $t_2[\vartheta_2]$ produces an empty net-token, puts it on the inhibitor place $p^i$, and puts a token back on the run-place.

The important aspect is, that the resulting marking $\mu$ does not correspond to a marking of $N^*$ (i.e. there is no $m$ such that $\mu = \tilde{\mu}(m)$ holds), since at least one control-place is marked with a non-empty token. And even more important, we can never get rid of these tokens again, since the tokens in the net-token on a control-place are never removed.

So, we have that all the net-tokens on control-places have the empty marking if and only if all guesses on the emptiness of inhibitor places have been right during the simulation: When all guesses have been right during the simulation then the resulting marking perfectly reflects the marking $m$. But after the first wrong guess we never reach a marking $\mu$ such that it is a configuration marking $\tilde{\mu}(m)$ for some $m$, since we can never get rid of the tokens which mark the net-token on some place $\mathsf{control}(p)$. □

From Lemma 4.3 undecidability of the reachability problem for conservative Eos easily follows (see Theorem 4.5 below). Before we give a reduction from the reachability problem for inhibitor nets to the liveness problem for conservative Eos. Again the constructions from Chapter 3.3 can be reused (see Lemma 3.15).

**Lemma 4.4.** *The reachability problem for inhibitor nets is reducible to the liveness problem for conservative* Eos.

*Proof.* Figure 4.2 shows the conservative Eos $OS_c(N^*)$ with the following property: If one can decide liveness for $OS_c(N^*)$, then one can decide reachability of the empty marking in the inhibitor net $N^*$. The construction is quite similar to the one in Lemma 3.15: In addition to the construction given there, we add the transitions $t_1^{(p)}$ and $t_2^{(p)}$ to each control place $\mathsf{control}(p)$. Whenever $t_1^{(p)}[\vartheta_1]$ with $\vartheta_1(N(p)) = \{\mathsf{d}_{N(p)}\}$ is enabled, the net-token is not empty. Note that the sequence $t_1^{(p)}[\vartheta_1] \cdot t_2^{(p)}[\vartheta_2]$ with $\vartheta_2(N(p)) = \{\mathsf{i}_{N(p)}\}$ does not change the marking.

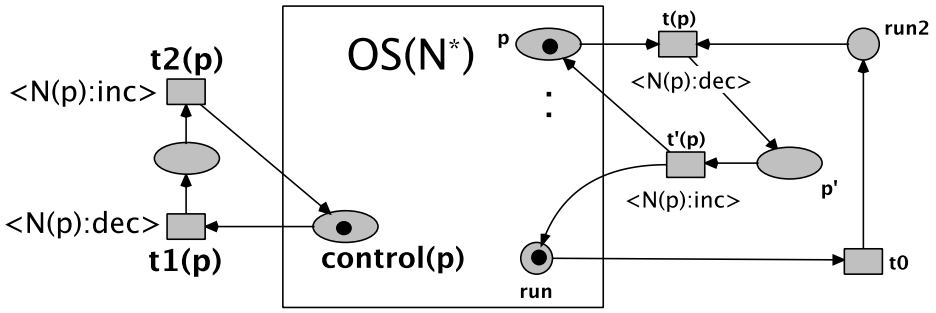


Figure 4.2: Conservative Eos-Reduction from **0**-Reachability to Liveness

As before non-liveness of $t_0[\vartheta_0]$ with $\vartheta_0 = \mathbf{0}$ indicates reachability of the empty marking $m = \mathbf{0}$ in the inhibitor net $N^*$ under the the assumption that all guesses have been made correctly. So, we have to express the condition that all guesses about inhibitor tests have been right during the weak simulation in terms of liveness: If all guesses about inhibitor tests on place $p \in P^i$ have been right during the weak simulation, then $t_1^{(p)}[\vartheta_1]$ is dead. Conversely, if one guess has been wrong, then $t_1^{(p)}[\vartheta_1]$ is live. Therefore, the simulation is correct iff $t_1^{(p)}[\vartheta_1]$ is dead for all $p \in P^i$.

Therefore, the empty marking $\mathbf{0}$ is reachable in the inhibitor net $N^*$ iff for all $p \in P^*$ we have that $t_1^{(p)}[\vartheta_1]$ with $\vartheta_1(N(p)) = \{\mathsf{d}_{N(p)}\}$ and $t_0[\vartheta_0]$ with $\vartheta_0 = \mathbf{0}$ are not live. □

From Lemma 4.3 and 4.4 we deduce that reachability and liveness are undecidable for conservative Eos.

**Theorem 4.5.** *Reachability and liveness are undecidable for conservative* Eos.

*Proof.* Since reachability is undecidable for inhibitor nets, undecidability of the liveness problem for Eos follows instantly from Lemma 4.4.

For reachability we use the construction of Lemma 4.3 to reduce a given inhibitor net $N^*$ to a conservative Eos $OS_c(N^*)$ and a given marking $m$ of $N^*$ of the inhibitor net to the marking $\tilde{\mu}(m)$ of $OS_c(N^*)$.

If $m$ is reachable in the inhibitor net, $\tilde{\mu}(m)$ is also reachable in $OS_c(N^*)$ by Lemma 4.3. Conversely, if the marking $\tilde{\mu}(m)$ is reachable in the conservative Eos, then in the firing sequence from the initial marking to $\tilde{\mu}(m)$ only

markings that correspond to markings of the inhibitor net appear and thus by construction a corresponding firing sequence in the inhibitor net occurs and thus $m$ is reachable in $N^*$. This shows the correctness of the reduction and thus undecidability of the reachability problem for conservative Eos. □

Although boundedness and coverability are decidable for conservative Eos, the undecidability of important problems like liveness and most notably reachability are undesirable if one attempts to verify properties of a model.

In the next section the formalism is therefore restricted further to generalised state machines.

## 4.2 Generalised State Machines

While conservative Eos are restricted in such a way that net tokens can neither be created nor destroyed, they can still by "joined" or "split", that is, there might be, for example, two places typed with the same net token type in the preset of a system net transition $\widehat{t}$ and only one place of this type in its postset. The markings of the two net tokens in the preset of $\widehat{t}$ are then added together and constitute the new marking of the net token in the postset of $\widehat{t}$ (if a synchronous event happens the effect is slightly different, but the main idea remains the same).

If conservative Eos are restricted further in such a way that this is prevented, the class of *generalised state machines* (GSMs) is obtained.

A *generalised state machine* (GSM) first introduced in [KR05] (there called *ordinary object-net systems*) is an Eos such that every system net transition has either exactly one place in its preset and one in its postset typed with the same object net or there are no such places. Additionally, the initial marking has at most one net-token of each type.

Although Eos are more powerful from a modelling point of view, in many cases certain aspects of them are not needed. Generalised state machines retain the ability to describe nesting of objects, synchronisation, and mobility, but the creation or destruction of objects is not allowed and neither is is allowed to split the inner marking of a net token or to join the inner markings of several. Thinking of these inner markings as representing the inner state of an object or agent, this is a reasonable restriction and GSM are then nicely suited to model physical entities.

Moreover, while Eos are an expressive modelling tool, the formalism is also Turing-complete, prohibiting automatic verification. For generalised state machines, on the other hand, it is possible to construct a p/t net that simulates its behaviour and thus everything that is decidable for p/t nets is also decidable for generalised state machines (cf. [KR05]).

**Definition 4.6** (**Generalised State Machines**). *Let* $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *be an* Eos. $G$ *is a* generalised state machine *(GSM) iff for all* $N \in \mathcal{N} \setminus \{N_\bullet\}$

1. $\forall \widehat{t} \in \widehat{T} : |\{\widehat{p} \in {}^\bullet\widehat{t} \mid d(\widehat{p}) = N\}| = |\{\widehat{p} \in \widehat{t}^\bullet \mid d(\widehat{p}) = N\}| \leq 1$
2. $\sum_{\widehat{p} \in \widehat{P}, d(\widehat{p}) = N} \Pi^1(\mu_0)(\widehat{p}) \leq 1$

*holds.*

Note that if the second item holds, $\sum_{\widehat{p} \in \widehat{P}, d(\widehat{p}) = N} \Pi^1(\mu)(\widehat{p}) \leq 1$ will hold for every reachable marking $\mu$ due to the first item. This can be easily proven by induction.

Since there is no restriction on $N_\bullet$, each p/t net is also a GSM. Moreover, for each GSM $G$ a p/t net, the reference net $\textsc{Rn}(G)$, can be easily constructed (see [KR05]). It is obtained by taking as set of places the disjoint union of all places of $G$ and as set of transitions the events of $G$. Since the places of all nets in $\mathcal{N}$ are disjoint, given a marking $\mu$ of $G$, the projections $(\Pi^1(\mu), (\Pi^2_N(\mu))_{N \in \mathcal{N}})$ can be identified with the multiset

$$\textsc{Rn}(\mu) := \Pi^1(\mu) + \sum_{N \in \mathcal{N}} \Pi^2_N(\mu),$$

which is a marking in the reference net.

**Definition 4.7** (**Reference Net**). *Let* $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *be a GSM. The* reference net*, denoted by* $\textsc{Rn}(G)$*, is defined as the p/t net:*

$$\textsc{Rn}(G) = \left(\left(\widehat{P} \cup \bigcup\nolimits_{N \in \mathcal{N}} P_N\right), \Theta, \mathbf{pre}^{\textsc{Rn}}, \mathbf{post}^{\textsc{Rn}}, \textsc{Rn}(\mu_0)\right)$$

*where* $\mathbf{pre}^{\textsc{Rn}}$ *and* $\mathbf{post}^{\textsc{Rn}}$ *are defined for an event* $\widehat{t}[\vartheta]$ *by:*

$$\begin{aligned} \mathbf{pre}^{\textsc{Rn}}(\widehat{t}[\vartheta]) &= \mathbf{pre}(\widehat{t}) + \sum\nolimits_{N \in \mathcal{N}} \mathbf{pre}_N(\vartheta(N)) \\ \mathbf{post}^{\textsc{Rn}}(\widehat{t}[\vartheta]) &= \mathbf{post}(\widehat{t}) + \sum\nolimits_{N \in \mathcal{N}} \mathbf{post}_N(\vartheta(N)), \end{aligned}$$

*with* $\mathbf{pre}(\widehat{\epsilon}) = \mathbf{post}(\widehat{\epsilon}) = \mathbf{0}$ *and* $\mathbf{pre}_N(\epsilon) = \mathbf{post}_N(\epsilon) = \mathbf{0}$ *for all* $N \in \mathcal{N}$.

The term *reference net* stems from the fact that $\textsc{Rn}(G)$ behaves as if each object net (in $G$) would have been accessed via pointers and not like a value. Since each object-net exists in a GSM at most once, the difference between references and values does not truly exist, cf. [KR05] where it is proven that reference and value semantics are equivalent for GSMs. We still use the term reference net, since the above definition can also be used for Eos, for which this difference does exist, but for which Theorem 4.8 below in general only holds in one direction.

We repeat two easy to prove theorems (cf. [Köh07] and [KR05]) which allow to carry over results for p/t nets to generalised state machines:

**Theorem 4.8.** *Let $G$ be a generalised state machine. An event $\widehat{t}[\vartheta]$ is activated in $G$ for $(\lambda, \rho)$ iff it is in* $\text{RN}(G)$*:*

$$\mu \xrightarrow[G]{\widehat{t}[\vartheta](\lambda,\rho)} \mu' \quad \Longleftrightarrow \quad \text{RN}(\mu) \xrightarrow[\text{RN}(G)]{\widehat{t}[\vartheta]} \text{RN}(\mu')$$

**Theorem 4.9.** *The reachability, liveness, coverability, and boundedness problems are decidable for generalised state machines.*

Although these problems and many others are decidable for GSMs they are at least as hard to decide as for p/t nets, since even a strongly deterministic GSM (see below) can be seen as a generalisation of a p/t net, and thus many of them are ExpSpace-hard (see Theorems 2.37 and 4.16). It is therefore natural to ask if certain structural or dynamic restrictions known from p/t nets can be carried over – maybe with adaptions – to generalised state machines or if new ones can be found and if these restrictions help to solve certain problems quicker. In the next section we will focus on structural restrictions of GSMs and the reachability problem for these restrictions. In Chapter 5 we will focus on dynamic restrictions especially on safeness.

## 4.3 Deterministic and Strongly Deterministic GSMs

While problems like reachability and many others are decidable for generalised state machines (see Theorem 4.9), the borderline mentioned in the introduction is not well understood for GSMs, i.e. what is an appropriate net class for the nets involved such that we enjoy a high modelling capability, yet also have the possibility to verify the models with affordable resources? What structural, dynamic or other restrictions are helpful to arrive at that goal?

To understand this boundary better, we first restrict GSMs to deterministic and strongly deterministic GSMs in this section, limiting the interaction between the involved nets. In the following section, we then severely restrict the nets to P- and T-nets. While such nets considered individually are well understood (cf. [DE95]), viewed in the context here their analysis becomes far more intricate.

In the subsequent sections, we then discuss further structural restrictions which where mentioned before in Chapter 2 for p/t nets. We will introduce acyclic generalised state machines, conflict-free generalised state machines and free-choice generalised state machines and focus on the reachability problem for these formalisms.

Although for every GSM the reference net can be constructed and thus problems like reachability can be decided, a major problem is that the reference net might become huge even for small GSMs. This is due to the fact that the *set of events* is present in the definition of the reference net (Definition 4.7).

**Lemma 4.10.** *Let $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a GSM and $|T| := \max\{|T_N| \mid N \in \mathcal{N}\}$. Then $|\Theta|$, i.e. the number of events of $G$, is bounded above by $|\widehat{T}| \cdot |T|^{|\mathcal{N}|}$.*

*Proof.* We sketch a GSM $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ with an exponential number of events in the size of $G$. Let $T_i$ be the set of transitions of the object net $N_i \in \mathcal{N}$. Let $\widehat{l}(\widehat{t})(N_i) = c_i$ for all $i$ and one system net transition $\widehat{t}$, where the $c_i$ are channels. Let $l_{N_i}(t) = c_i$ for all $t \in T_i$ and all $i$. Now $\widehat{t}$ may fire synchronously with each transition in $N_1$, each in $N_2$ and so on. Each of these possibilities results in a different event, so we already have at least $|T_1| \cdot |T_2| \cdot \ldots \cdot |T_n|$ events, a number exponential in the number of object nets and thus in the size of the GSM. Note that this is possible for each system net transition resulting in an even larger number of events and in the worst case in the bound above. The number can not grow larger because other labelings would decrease the number of possible nondeterministic choices and so the number of events. □

Given a GSM it might thus be very expensive to construct its reference net. Note that this is due to the nondeterminism introduced above by the labelling. All transitions of one object net are labelled with the same channel, so one of the transitions is chosen nondeterministically to fire synchronously with $\widehat{t}$. To prevent this, we introduced deterministic GSMs in [HKB11a]. Here we generalise this definition to Eos:

**Definition 4.11 (Deterministic and Strongly Deterministic Eos and GSMs).** *A Eos $OS$ is called* deterministic *if for each $N \in \mathcal{N}$ and every two transitions $t, t' \in T_N$, $t \neq t'$ with $l_N(t) \neq \tau \neq l_N(t')$, $l_N(t) \neq l_N(t')$ holds, i.e. if the labels for all all $t \in T_N$ with $l_N(t) \neq \tau$ are pairwise different.*

*$OS$ is* strongly deterministic *if $OS$ is deterministic and additionally for all $\widehat{t}$ and $N$ with $\widehat{l}(\widehat{t})(N) \neq \tau$ the labels $\widehat{l}(\widehat{t})(N)$ are pairwise different.*

Thus, in a deterministic Eos or GSM each channel is used at most once in each object net. In a strongly deterministic Eos or GSM each channel is also used at most once in the system net.

For Eos the definition of determinism does not significantly reduce the power of the formalisms introduced so far, namely of Eos or conservative Eos

**Theorem 4.12.** *For each inhibitor net $N^*$ there is a strongly deterministic* Eos *$OS_{strong}(N^*)$ that bisimulates $N^*$.*

*Proof.* The proof of Theorem 3.14 can be adjusted by adding some transitions and channels so that the resulting Eos is strongly deterministic.

Each object net $N(p)$, $p \in P^*$ does not only have one transition $i_{N(p)}$ to increase the number of tokens on $cnt_{N(p)}$ and one transition $d_{N(p)}$ to decrease them, but one for each transition $t \in T^*$, i.e. the place $cnt_{N(p)}$ has the transitions $\{i_{N(p),t} \mid t \in T^*\}$ in its preset and the transitions $\{d_{N(p),t} \mid t \in T^*\}$ in its postset. Instead of $inc_{N(p)}$ and $dec_{N(p)}$ the channels $inc_{N(p),t}$ and $dec_{N(p),t}$, $t \in T^*$ are

used, i.e. the transition $i_{N(p),t}$ is labelled with $inc_{N(p),t}$ and in the same manner for the transitions in the postset. The object nets' transitions then all use different channels.

In the system net the transitions $t_1$ and $t_2$ which originated from a transition $t \in T^*$ only use the channels with subscript $t$ now, i.e. if $t$ had a place $p_1$ in its preset, then $t$ synchronises with the object net $N(p_1)$ via the channel $dec_{N(p_1),t}$ and likewise for places in the postset. The system nets' transitions then all use different channels as well and the Eos is thus strongly deterministic but otherwise has the same behaviour as the net constructed in Theorem 3.14. □

**Theorem 4.13.** *For each inhibitor net $N^*$ there is a strongly deterministic conservative* Eos *$OS_c(N^*)$ satisfying the statements in Lemma 4.3*

*Proof.* The Eos constructed in the proof of Lemma 4.3 can be adjusted in the same way as in Theorem 4.12 resulting in a strongly deterministic and conservative Eos with the same behaviour. □

The above two theorems imply

**Corollary 4.14.** *The reachability problem for strongly deterministic, deterministic and general* Eos *as well as for strongly deterministic, deterministic and general (with regard to determinism) conservative* Eos *is undecidable.*

Turning to GSMs again, the bound on the number of events given in Lemma 4.10 can now be crucially decreased.

**Lemma 4.15.** *Let $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a deterministic or strongly deterministic GSM, then $|\Theta|$ is bounded above by $|\widehat{T}| + \sum_{N \in \mathcal{N}} |T_N|$.*

*Proof.* A GSM $G$ has at most $|\widehat{T}|$ system-autonomous and $\sum_{N \in \mathcal{N}} |T_N|$ object-autonomous events. If $G$ is deterministic, then a system net transition $\widehat{t} \in \widehat{T}$ can participate in at most one synchronous event, because if $\widehat{l}(\widehat{t})(N) = c$ for an object net $N \in \mathcal{N}$, then there is at most one transition in $N$ which may fire synchronously with $\widehat{t}$, because only one transition may be labelled with $c$ in $N$. Thus the sum of the number of system-autonomous and the number of synchronous events is bounded above by $|\widehat{T}|$ and thus the number of all events is bounded above by $|\widehat{T}| + \sum_{N \in \mathcal{N}} |T_N|$.

Since strongly deterministic GSMs are a further restriction of deterministic GSMs, this argument holds for them, too. □

While for a GSM $G$ the number of events might thus be exponential in the size of $G$, this number is only polynomial in the size of $G$, if $G$ is deterministic or strongly deterministic. However, even a strongly deterministic GSM does not reduce the complexity in comparison with p/t nets, yet.

**Theorem 4.16.** *Every p/t net system $\mathcal{N}$ can be simulated by a strongly deterministic GSM $G_{\mathcal{N}}$.*

*Proof.* Let $\mathcal{N} = (N, m_0)$ be a p/t net system. The GSM $G_{\mathcal{N}}$ can be obtained by using $N$ as system net, typing all places with $N_\bullet$, which is the only object net present. The system net transitions are all labelled with $\tau$ for system autonomous firing. The initial marking $\mu_0$ is obtained from $m_0$ by augmenting each place marked in $m_0$ by the submarking $\mathbf{0}$, i.e. from $m_0 = \sum_i p_i$ follows $\mu_0 = \sum_i p_i[\mathbf{0}]$. The main difference between $\mathcal{N}$ and $G_{\mathcal{N}}$ is that the latter uses the object net $N_\bullet$ which is simply a net without places and transitions and thus does not synchronise. □

**Corollary 4.17.** *The reachability problem for strongly deterministic, deterministic and general GSMs is* ExpSpace*-hard.*

On the one hand GSMs can thus be used to model mobility and communication of processes or agents - even if only to a smaller degree compared to Eos or general object nets. On the other hand a GSM $G$ can be "flatten" to the reference net Rn($G$), which is a p/t net and which thus makes it possible to use analyses techniques for p/t nets. Unfortunately, this p/t net can be very big compared to the size of the GSM. While determinism and strongly determinism as introduced above counteract this blow-up, the resulting GSMs are still as powerful as p/t nets and it is now interesting to investigate if certain restrictions known for p/t nets can be transferred to GSMs and if they retain their complexity.

In the following sections, we focus on P- and T-nets, on acyclic nets, on conflict-freedom and on the free-choice property.

## 4.4 P- and T-nets for Generalised State Machines

While determinism and strong determinism are already restrictions of generalised state machines, we further restrict the nets involved to P- and T-nets (see Definition 2.40) here.

Although GSMs with these restrictions are likely to be of little use in modelling applications, understanding them better might help in future attempts to analyse more sophisticated formalisms.

For P- and T-nets reachability and liveness can be quickly decided (see Theorem 2.41). It turns out that this is not the case for GSMs. We show that, given a strongly deterministic GSM where the system net and all object nets are P-nets, it is PSpace-complete to decide the reachability of a given marking. We show that for the same restriction to T-nets, the reachability problem remains solvable in polynomial time. We then turn to combinations of restricting the

system and/or the object nets to P- and/or T-nets, give some first results, and discuss the problems that arise with these cases. Throughout we also discuss the effect of dropping the restriction to strongly deterministic GSMs for these formalisms. The results obtained have been published in [HKB12b] and are summarised in Table 4.1.

**Definition 4.18.** *Let $N = (P, T, F)$ be a p/t net and $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a GSM.*

1. *If $|{}^\bullet t| = |t^\bullet| = 1$ holds for every transition $t \in T$, then $N$ is a* P-net.
2. *If $|{}^\bullet p| = |p^\bullet| = 1$ holds for every place $p \in P$, then $N$ is a* T-net.
3. *If $\widehat{N}$ is a P-net and all $N \in \mathcal{N}$ are P-nets, then $G$ is a* ppGSM.
4. *If $\widehat{N}$ is a T-net and all $N \in \mathcal{N}$ are T-nets, then $G$ is a* ttGSM.
5. *If $\widehat{N}$ is a P-net and all $N \in \mathcal{N}$ are T-nets, then $G$ is a* ptGSM.
6. *If $\widehat{N}$ is a T-net and all $N \in \mathcal{N}$ are P-nets, then $G$ is a* tpGSM.

*For historical reasons P-nets are also called* S-nets *and T-nets are also called* marked graphs.

The distinction sometimes made between a P-net $N$ and a *P-system* $(N, m_0)$, where additionally the initial marking $m_0$ is given (analogous for T-nets) is not use here. We simply speak of a P- or T-net. It will be clear from the context or stated explicitly if only the net structure is meant.

From the definition of a GSM and the restrictions imposed above, one can deduce that for ppGSMs and ptGSMs there can only be one object net:

**Lemma 4.19.** *If $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ is a ppGSM or a ptGSM and the system net $\widehat{N}$ is connected, then $|d(\widehat{P})| = 1$.*

*Proof.* Assume otherwise and let $D$ be the set of tuples of system net places such that $(\widehat{p}, \widehat{p}') \in D$ iff $d(\widehat{p}) \neq d(\widehat{p}')$. Let $dist(\widehat{p}, \widehat{p}')$ be the distance of two system net places with respect to the relation $(\widehat{F} \cup \widehat{F}^{-1})^*$. Choose $(\widehat{p}_1, \widehat{p}_2) \in D$ such that $dist(\widehat{p}_1, \widehat{p}_2)$ is minimal.

Now, since $\widehat{N}$ is connected and since $dist(\widehat{p}_1, \widehat{p}_2)$ is minimal, there is a node $x$ such that a directed path (possibly of length 0) with respect to the relation $\widehat{F}$ exists from $\widehat{p}_1$ to $x$ and also from $\widehat{p}_2$ to $x$. (If no such $x$ exists, there is a node $x'$ on the path between $\widehat{p}_1$ and $\widehat{p}_2$ such that $(\widehat{p}_1, x') \in D$ or $(\widehat{p}_2, x') \in D$ holds and such that the distance is smaller than $dist(\widehat{p}_1, \widehat{p}_2)$, contradicting the choice of $\widehat{p}_1$ and $\widehat{p}_2$.) Since $G$ is a ppGSM or a ptGSM, $|{}^\bullet\widehat{t}| = |\widehat{t}^\bullet| = 1$ holds for all $\widehat{t} \in \widehat{T}$ and thus $x$ cannot be a transition and must be a place. But now, since every transition of the system net has exactly one place in its preset and

one in its postset and since these must then be of the same type according to the definition of a GSM (item 1 in Definition 4.6), $d(x)$ must equal $d(\widehat{p}_1)$ due to the path from $\widehat{p}_1$ to $x$ but must also equal $d(\widehat{p}_2)$ due to the path from $\widehat{p}_2$ to $x$, contradicting $d(\widehat{p}_1) \neq d(\widehat{p}_2)$. □

We assume that $\mathcal{N} = d(\widehat{P})$ in the following, since a $N \in \mathcal{N} \setminus d(\widehat{P})$ is never used. The above lemma thus says that in a connected ppGSM or ptGSM we always have $|\mathcal{N}| = 1$. Furthermore, we may usually assume connectedness in the algorithms below, because if a GSM is not connected the single parts do not affect each other and can be treated in isolation. Since in the case where $\mathcal{N} = \{N_\bullet\}$ holds, ppGSMs and ptGSMs are P-Nets, ttGSMs and tpGSMs are T-nets, and GSMs are standard p/t nets, and thus the theory for P-nets, T-nets, and general p/t nets is applicable, it is reasonable to exclude the case $\mathcal{N} = \{N_\bullet\}$ for all GSMs. We assume in the following that $\mathcal{N} \neq \{N_\bullet\}$ for GSMs. Also note that if $|\mathcal{N}| = 1$ holds for a GSM $G$ and $\mu_0 \neq \mathbf{0}$ holds, then $G$ has exactly one object net in every reachable marking because the initial marking has only one net-token of each type (see the second item in the definition of a GSM, Definition 4.6).

To sum up, we assume in the following that a given GSM $G$ is connected, that $\mathcal{N} = d(\widehat{P})$, and that $\mathcal{N} \neq \{N_\bullet\}$ holds.

In the following, we will focus on the complexity of the reachability problem for strongly deterministic GSMs which additionally fulfil one of the items 3 to 6 in Definition 4.18 above.

### ppGSMs.

While for P- and T-nets the reachability problem can be solved in polynomial time [DE95], the problem becomes more intricate in the context of ppGSMs or ttGSMs. It turns out that for ttGSMs reachability can still be decided in polynomial time, but for ppGSMs the problem becomes PSpace-complete. We start with showing PSpace-hardness for the last-mentioned problem and then continue to show three possibilities to decide the problem in polynomial space. One of these approaches can also be used to prove that for ttGSMs reachability of a given marking can be decided in polynomial time.

**Lemma 4.20.** *Given a strongly deterministic ppGSM $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ and a marking $\mu$, it is* PSpace*-hard to decide if $\mu$ is reachable from $\mu_0$.*

*Proof.* We give a reduction from the problem to decide if, given a linear bounded automaton $A$ and an input string $x$, whether $A$ accepts $x$ or not. This problem is PSpace-complete [Kar72], [GJ79].

Let $A = (Q, \Sigma, \Gamma, K, q_0, F, \#)$ be a linear bounded automaton where $Z$ is the finite set of states, $\Sigma$ the finite set of input symbols, $\Gamma \supseteq \Sigma \cup \{\#\}$ the finite set of tape symbols, $K \subseteq Q \times \Gamma \times \{L, R, H\} \times Q \times \Gamma$ the transition relation, $q_0$ is

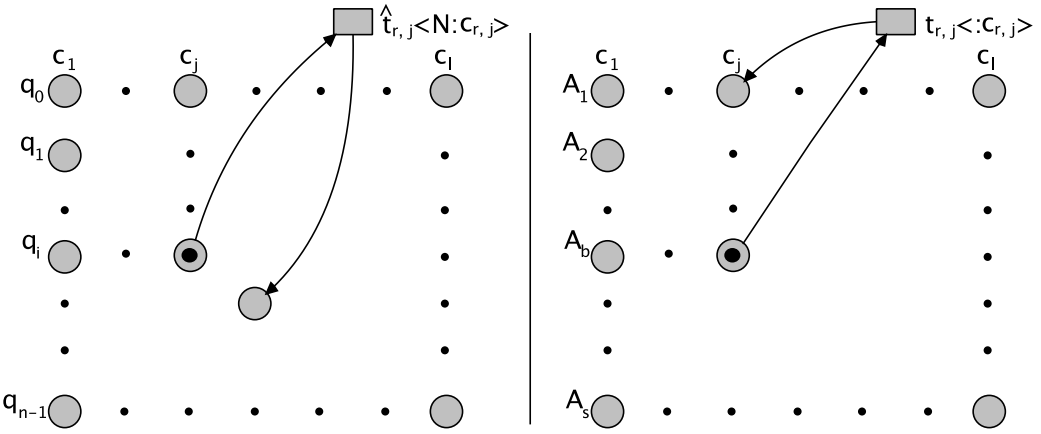


Figure 4.3: System net (left) and object net (right) of the ppGSM from Lemma 4.20.

the initial state, $F$ the set of final states and # the blank symbol. Without loss of generality we assume that $A$ uses only the portion of the tape containing the input (due to the linear tape-compression theorem) and that if $A$ accepts a word it clears the tape, moves the head to the leftmost cell and enters a unique final state (i.e. we also assume $|F| = 1$).

Given an LBA $A$ and an input string $x \in \Sigma^*$, we now construct in polynomial time a strongly deterministic ppGSM $G$ and a marking $\mu$ such that $\mu$ is reachable in $G$ iff $A$ accepts $x$.

The idea is to have an event $\theta$ for every transition $k \in K$ *and* every tape cell on which this transition $k$ might occur. Tokens on places are used to memorize in which state $A$ is, on which cell the read-/write-head resides and what tape symbol is written on each cell. Since a single transition with only one input and one output arc cannot do all these changes the information is partly stored in the system and partly stored in the object net. The system net will save in which state $A$ is and on which cell the read-/write head currently resides. The object net will save for each cell used what tape symbol is written on it (see Figure 4.3).

Let $Q = \{q_0, \ldots, q_{n-1}\}$, where $q_0$ is the start and $q_1$ the final state, $K = \{k_1, \ldots, k_m\}$, $\Gamma = \{A_1, \ldots, A_s\}$, where $A_1 = \#$, $|x| = l$ the input string's length and $c_1, \ldots, c_l$ the tape cells used.

The system net consists of $n \cdot l$ places $\widehat{P} = \{\widehat{p}_{i,j} \mid 0 \leq i \leq n-1, 1 \leq j \leq l\}$, where $p_{i,j}$ is marked if $A$ is in state $q_i$ and the read-/write-head is on cell $c_j$.

The (single) object net consists of $s \cdot l$ places $P = \{p_{i,j} \mid 1 \leq i \leq s, 1 \leq j \leq l\}$, where $p_{i,j}$ is marked if the symbol $A_i$ is written on the cell $c_j$.

Furthermore, for each cell $c_j$ and each transition $k_i \in K$ of the LBA there is a system net transition $\widehat{t}_{i,j}$ and an object net transition $t_{i,j}$ which are labelled

with the same channel $c_{i,j}$. (The cases where the read-/write-head moves from the tape are discussed below.) Let $k_i = (q_a, A_b, X, q_{a'}, A_{b'})$. The input arc of $\widehat{t}_{i,j}$ is from $\widehat{p}_{a,j}$, the place that encodes that $A$ is in state $q_a$ and the read-/write-head is on cell $c_j$. The input arc of $t_{i,j}$ is from $p_{b,j}$, the place that encodes that the cell $c_j$ is currently marked with the tape symbol $A_b$. The output arc of $t_{i,j}$ is $p_{b',j}$. At last, the output arc of the system net transition depends on $X$, the movement of the LBA's head. If $X = H$, then the output arc of $\widehat{t}_{i,j}$ is to $p_{a',j}$. If $X = L$, then the output arc is to $p_{a',j-1}$ and if $X = R$ the output arc is to $p_{a',j+1}$. In the cases where $X = L$ and $j = 0$ or $X = R$ and $j = l$, i.e. in the cases where the LBA's head would move off the tape no transitions exist. The situation in Figure 4.3 is therefore as follows: The LBA is in state $q_i$, reading cell $c_j$ onto which currently symbol $A_b$ is written. The content of the other cells is not shown in the figure. The pictured transitions - synchronised via the channel $c_{r,j}$ - correspond to a transition $k_r = (q_i, A_b, R, q_{i+1}, A_1)$ of $A$.

Since we have only one object net, all places of the system net are typed with this net.

For the initial marking $\mu_0$ let $x = A_{i_1} A_{i_2} \dots A_{i_l}$ be the input of length $l$. Now $\mu_0$ is given by $\widehat{p}_{0,1}[p_{i_1,1} + p_{i_2,2} + \dots + p_{i_l,l}]$.

The marking tested for reachability is given by $\mu_e := \widehat{p}_{1,1}[p_{1,1}+p_{1,2}+\dots+p_{1,l}]$.

By construction exactly one system net place is marked in each reachable marking. Also from the object net's places $p_{1,i}, p_{2,i}, \dots, p_{s,i}$ exactly one place is marked (and $l$ places are marked altogether in the object net).

Now, if $A$ accepts the input $x$, then there is a finite sequence $C_1, C_2, \dots$ of configurations such that for each $C_i$ there is a transition $k_i \in K$ that is possible in $C_i$ and that changes the configuration of $A$ to $C_{i+1}$. It is easy to prove inductively that this sequence of configurations corresponds to a sequence of markings in the constructed net system and that the transition $k_i \in K$ correspond to exactly one event consisting of one system net and one object net transition. Firing this event yields the marking corresponding to the next configuration in the sequence. If $A$ accepts $x$, then the last configuration is the unique accepting configuration and so the reached marking in the net system is $\mu_e$.

Conversely, if $\mu_e$ is reachable in $G$, then by construction the sequence of markings correspond to a sequence of configurations of $A$ and each transition in $G$ corresponds to a transition in $A$. (A more rigorous proof would again use induction.) Thus if $\mu_e$ is reachable in $G$, so is the accepting configuration in $A$ and hence $\mu_e$ is reachable in $G$ iff $A$ accepts $x$.

The constructed net system is clearly a strongly deterministic ppGSM. Furthermore, the construction can be done in polynomial time: With $s := \max\{|Q|, |\Gamma|, |K|, |x|\}$ we have $|\widehat{P}|, |P|, |\widehat{T}|, |T| \leq s^2$ and $|\widehat{F}|, |F| \leq 2s^2$. The labelling assigns to each transition one channel and can thus be stored in $O(s^2)$ space. Also $\mu_0$ and $\mu_e$ can be stored in $O(s)$ space. Altogether the output is in $O(s^2)$ space and since only a constant amount of computation is necessary for

each output bit, the whole computation is possible in $O(s^2)$ time and actually even in logarithmic space on the work tape. We conclude that the reachability problem for ppGSMs is PSPACE-hard. □

The negative result above carries over to deterministic ppGSMs and general ppGSMs, thus the reachability problem for all of them is at least PSPACE-hard. We now prove that PSPACE is enough.

In our first approach to prove that the reachability problem for ppGSMs can be decided in polynomial space we use a technique that dates back to Savitch's proof of PSPACE = NPSPACE [Sav70]. The technique is based on the following idea: Given a finite state space of size $2^{p(n)}$ where $p$ is a polynomial and $n$ the input size, to decide if a given state $s$ is reachable from the start state $s_0$ in $m$ steps a deterministic machine iterates through all other states $s'$ and tests if $s'$ is reachable from $s_0$ and $s$ from $s'$ by $m/2$ steps, i.e. by half the steps. These tests are done recursively applying the same technique and reusing space. Since the number of steps is halved each time and at most $2^{p(n)}$ steps are possible without entering a loop, only $\log 2^{p(n)} = p(n)$ states need to be stored on a stack and thus the space needed is polynomial.

This technique works here, too, even if the state space might be rather big.

**Lemma 4.21.** *Given a strongly deterministic ppGSM $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ and a marking $\mu$, it is decidable in polynomial space if $\mu$ is reachable from $\mu_0$.*

*Proof.* Let $N = (P, T, F)$ be the sole object net (see Lemma 4.19). If $\mu_0 = \mathbf{0}$, then the only reachable marking is $\mathbf{0}$, since $\widehat{N}$ is a P-Net and thus no transition is active. We thus assume that $\mu_0 \neq \mathbf{0}$. In this case $N$ is not only the sole object net type, but actually there is only one object net in any reachable marking $\mu$ due to the second item in Definition 4.6, i.e. $|\Pi^1(\mu)| = 1$.

Let $|\Pi^2_N(\mu_0)| = m$, i.e. in the initial marking the object net is marked with $m$ black tokens. Since $N$ is a P-net, the number of tokens in $N$ remains constant. On each place between 0 and $m$ tokens can reside, thus the number of reachable markings of $N$ is bounded by $(m+1)^{|P|}$. This bound could be strengthened, but is sufficient here. Since $N$ can move around in the system net, but only one place of the system net is marked at any reachable marking, we have an upper bound of $|\widehat{P}| \cdot (m+1)^{|P|}$ for the number of reachable markings of $G$. Let $n$ be the size of the input. From $n > \log(m+1), |P|, |\widehat{P}|$ and $|\widehat{P}| \cdot (m+1)^{|P|} = 2^{\log|\widehat{P}|} \cdot 2^{\log(m+1)\cdot|P|} \leq 2^{n^2+n}$ it follows that we have a finite state space of size $2^{p(n)}$ where $p$ is a polynomial and $n$ the input size and thus the technique outlined above is applicable.

A NPSPACE-Algorithm $A$ works as follows: Given $G$, $\mu_0$ and $\mu$ we first guess the number $i$ of steps it takes to reach $\mu$. Since we have an upper bound for the size of the state space, $i$ is known to be between 0 and $2^{n^2+n}$.

Instead of guessing $i$ at the beginning, it is also possible to implement a loop, using $i$ as a loop index and reusing space in each iteration.

Now $A$ guesses a marking $\mu'$ and verifies recursively that $\mu'$ is reachable from $\mu_0$ and $\mu$ from $\mu'$ with $j = \lceil i/2 \rceil$ resp. $j = \lfloor i/2 \rfloor$ steps. The recursion ends if $j$ is 0 or 1. In the first case two markings have to be tested for equality. In the second case we have to test if one marking is reachable from the other in one step. For this we can iterate through all events, consistently reusing space, test if the event is active and, if so, test if it has the desired effect (see Lemma 3.11). $A$ accepts if it reaches $\mu$ and rejects if it does not reach $\mu$ in $i$ steps.

Since the nesting depth of the recursive calls is $\log i$, it is at most $\log 2^{n^2+n} = n^2 + n$ and thus polynomial in the input size. Furthermore, since only a finite number of data items (e.g. markings, counters) need to be stored and all these only require polynomial space and since it is possible to test in polynomial space if a marking is reachable from another marking and also if a marking is identical to another, i.e. all subroutines only require polynomial space, polynomial space is sufficient for the whole computation. □

The technique used in the proof above was also used successfully in the proof that CTL model checking of 1-safe p/t nets is possible in PSpace [Esp98a]. A result that can be generalised to safe Eos and also to safe Eos with an arbitrary but fixed nesting depth and to strongly safe object net systems, see Chapters 5 and 6.

Alternatively to the proof above, one can exploit the fact that the total number of tokens does not change in a ppGSM. This idea was also used in [JLL77] to prove that the reachability problem can be decided in polynomial space for 1-conservative p/t nets (i.e. nets with $|{}^\bullet t| = |t^\bullet|$ for all $t$). We only sketch the proof here.

**Alternative Proof of Lemma 4.21 (Sketch).** Since $\widehat{N}$ and the sole object net $N$ are both P-nets, the total number of tokens does not change. In a nondeterministic algorithm $A$ one can thus maintain one counter for each place, where the size of each counter is bounded. By guessing a firing sequence $A$ can thus solve the reachability problem again exploiting the fact that the state space is finite and the firing sequence has a length representable in polynomial space. The whole algorithm works in polynomial space. □

Both approaches presented above are working directly on a given GSM $G$. In the next approach we make use of the reference net $\text{Rn}(G)$ instead (see Definition 4.7), which allows us to use tools already available for p/t nets.

**Lemma 4.22.** *Let $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a strongly deterministic ppGSM. Then the p/t net $\text{Rn}(G)$ satisfies $|{}^\bullet t| = |t^\bullet| = 1$ or $|{}^\bullet t| = |t^\bullet| = 2$ for all $t \in T(\text{Rn}(G))$.*

*Proof.* Let $t \in T(\text{Rn}(G))$ and let $N$ be the unique object net (see Lemma 4.19). Note that $N$ is not only the sole object net type present but that actually only

one object net exists in any reachable marking (if $\mu_0 \neq \mathbf{0}$; but otherwise $\mathbf{0}$ is the only reachable marking), due to the second item in Definition 4.6.

We want to show $|\mathbf{pre}^{\mathrm{RN}}(t)| = |\mathbf{post}^{\mathrm{RN}}(t)| \in \{1, 2\}$. Since the transitions of $\mathrm{RN}(G)$ are the events of $G$ we distinguish the three cases $t \in \Theta_s$ (system-autonomous event), $t \in \Theta_o$ (object-autonomous event), and $t \in \Theta_l$ (synchronous event).

The idea is that in the first two cases of autonomous events the pre- and postset of the event are identical to the pre- resp. postset (of cardinality 1) of a single system or object net transition and that in the case of a synchronous event the event is just a combination of one system net and one object net transition so that the cardinality of the pre- and postset is 2.

Explicitly, we have $t = \widehat{t}[\vartheta_\epsilon]$ for some system net transition $\widehat{t}$ in the first case. Since $\vartheta_\epsilon(N) = \epsilon$, the preset of $t$ and $\widehat{t}$ are identical according to Definition 4.7 as are the postsets. Thus we have $|\mathbf{pre}^{\mathrm{RN}}(t)| = |\mathbf{pre}(\widehat{t})| = 1 = |\mathbf{post}(\widehat{t})| = |\mathbf{post}^{\mathrm{RN}}(t)|$.

The case of an object-autonomous event is similar. We then have $t = \widehat{\epsilon}[\vartheta]$ with $\vartheta(N) \neq \epsilon$ (usually for exactly one object net, but we only have one object net here, since $G$ is a ppGSM). Thus again according to Definition 4.7 we have that the preset of $t$ and $\vartheta(N)$ are the same as are the postsets and their cardinality is one again.

In the third case of a synchronous event one system net transition $\widehat{t}$ fires synchronously with exactly one object net transition $t_N$. With Definition 4.7 and since $\mathbf{pre}(\widehat{t}) \neq \mathbf{pre}_N(t_N)$ and $\mathbf{post}(\widehat{t}) \neq \mathbf{post}_N(t_N)$, we have $|\mathbf{pre}^{\mathrm{RN}}(t)| = |\mathbf{post}^{\mathrm{RN}}(t)| = 2$ (where in the preset of $t$ are exactly the two places in the preset of $\widehat{t}$ and $t_N$ and analogously for the postset). □

A p/t net with the property $|{}^\bullet t| = |t^\bullet|$ for each transition $t$ is called 1-conservative and it is shown in [JLL77] that for these net class the reachability problem can be decided in polynomial space. Thus, since the above conversion is clearly possible in polynomial space, we again have that reachability is decidable in polynomial space.

Note that we did not use the fact that $G$ was a strongly deterministic GSM in any of the above proofs and actually the proofs work all the same for deterministic ppGSMs and also for general ppGSMs. Thus we have the following corollary from Lemma 4.21 or Lemma 4.22 above:

**Corollary 4.23.** *The reachability problem for strongly deterministic ppGSMs, deterministic ppGSMs and ppGSMs is solvable in polynomial space.*

The main reason for the generalisation in Corollary 4.23 is that the exponential blow-up of events (see Lemma 4.10) cannot occur in a ppGSM – due to Lemma 4.19 there is only one object net and thus we have at most a quadratic number of events in the size of the input.

From Corollary 4.23 and from Lemma 4.20 we deduce the following theorem:

**Theorem 4.24.** *The reachability problem for strongly deterministic ppGSMs, deterministic ppGSMs and ppGSMs is* PSpace*-complete.*

### ttGSMs.

In a ttGSM the system net and all object nets are T-nets. Unlike before in the case of ppGSMs, a variety of object nets might now be present and according to Lemma 4.10 the number of events might become huge. Thus for general ttGSMs the approach from Lemma 4.22 is unlikely to be applicable. But for strongly deterministic ttGSMs the approach works. The proof of the following lemma first appeared in [HKB11b] and is similar in nature to the proof of Lemma 4.22. We show that the reference net $\mathrm{R}_N(G)$ is also a T-net, if $G$ is a strongly deterministic ttGSM. Since reachability can be decided in polynomial time for a T-net and since the construction of $\mathrm{R}_N(G)$ is possible in polynomial time in this context, we deduce that the reachability problem for strongly deterministic ttGSMs can be solved in polynomial time, too.

**Theorem 4.25.** *Let $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a strongly deterministic ttGSM. Then $\mathrm{R}_N(G)$ is a T-net.*

*Proof.* With $\mathbf{pre}_{\mathrm{R}_N}(p)$ and $\mathbf{post}_{\mathrm{R}_N}(p)$ we denote the preset and postset of a place $p$ in the reference net $\mathrm{R}_N(G)$. If $p$ is also a place in an object net $N \in \mathcal{N}$ we denote $p$'s pre- resp. postset in the original object net $N$ by $\mathbf{pre}_N(p)$, resp. $\mathbf{post}_N(p)$. The pre- and postset of a place $p$ in the system net $\widehat{N}$ are likewise denoted by $\mathbf{pre}_{\widehat{N}}(p)$ and $\mathbf{post}_{\widehat{N}}(p)$.

Now, let $p \in P(\mathrm{R}_N(G))$. Our goal is to show $|\mathbf{pre}_{\mathrm{R}_N}(p)| = |\mathbf{post}_{\mathrm{R}_N}(p)| = 1$. We distinguish two cases: $p \in \widehat{P}$ or $p \in P_N$ for a $N \in \mathcal{N}$. Let $p \in \widehat{P}$. Since $|\mathbf{post}_{\widehat{N}}(p)| = 1$ holds, $p$ is connected with a system net transition and thus associated with an event, thus $|\mathbf{post}_{\mathrm{R}_N}(p)| \geq 1$. Assume $|\mathbf{post}_{\mathrm{R}_N}(p)| > 1$ holds. Then two events $\widehat{t}[\vartheta], \widehat{t'}[\vartheta'] \in \mathbf{post}_{\mathrm{R}_N}(p)$ with $\widehat{t}[\vartheta] \neq \widehat{t'}[\vartheta']$ exist and $p \in \mathbf{pre}(\widehat{t}) \wedge p \in \mathbf{pre}(\widehat{t'})$ follows from Definition 4.7 and $p \in \widehat{P}$. This implies $\widehat{t} \neq \widehat{\epsilon} \neq \widehat{t'}$, since $\mathbf{pre}(\widehat{\epsilon}) = \mathbf{0}$.

The events are thus not object-autonomous. With this and from $\widehat{t}[\vartheta] \neq \widehat{t'}[\vartheta']$ we can deduce $\widehat{t} \neq \widehat{t'}$. Assume otherwise, i.e. assume $\widehat{t} = \widehat{t'}$. Then a net $N \in \mathcal{N}$ would have to exist with $\vartheta(N) \neq \vartheta'(N)$. But with $\vartheta(N) = t_1 \neq t_2 = \vartheta'(N)$ synchronisation would require that $l_N(t_1) = \widehat{l}(\widehat{t})(N) = l_N(t_2)$, which is not possible in a deterministic GSM, if $t_1, t_2 \in T_N$, because for a $t \in T_N$, $\vartheta(N) = t$ implies $l_N(t) \neq \tau$. If w.l.o.g. $t_1 \in T_N$ and $t_2 = \epsilon$, then $l_N(t_2) = l_N(\epsilon) = \tau = l_N(t_1)$ would mean that $t_1$ fires object-autonomously in contrast to $\widehat{t'} \neq \widehat{\epsilon}$. But with $\widehat{t} \neq \widehat{t'}$ and $p \in \mathbf{pre}(\widehat{t}) \wedge p \in \mathbf{pre}(\widehat{t'})$ we have a contradiction to $|\mathbf{post}_{\widehat{N}}(p)| = 1$.

The case $|\mathbf{pre}_{\mathrm{R}_N}(p)| > 1$ analogously leads to a contradiction and thus $|\mathbf{pre}_{\mathrm{R}_N}(p)| = |\mathbf{post}_{\mathrm{R}_N}(p)| = 1$ for $p \in \widehat{P}$.

Now let $p \in P_N$ for a $N \in \mathcal{N}$. Like above $|\mathbf{post}_{\mathrm{RN}}(p)|$ is at least 1. Assume $|\mathbf{post}_{\mathrm{RN}}(p)| > 1$. Again two events $\widehat{t}[\vartheta], \widehat{t'}[\vartheta'] \in \mathbf{post}_{\mathrm{RN}}(p)$ with $\widehat{t}[\vartheta] \neq \widehat{t'}[\vartheta']$ exist. This time $p \in \mathbf{pre}_N(\vartheta(N)) \wedge p \in \mathbf{pre}_N(\vartheta'(N))$ follows from Definition 4.7 and $p \in P_N$. Together with $\mathbf{pre}_N(\epsilon) = \mathbf{0}$ this implies $\vartheta(N) \neq \epsilon \neq \vartheta'(N)$. The events are thus not system-autonomous.

The inequality $\vartheta(N) \neq \vartheta'(N)$ immediately implies a contradiction to $|\mathbf{post}_N(p)| = 1$.

In the case of $\vartheta(N) = \vartheta'(N)$, neither $\widehat{t} = \widehat{t'} = \widehat{\epsilon}$, nor $\widehat{t} = \widehat{t'} \neq \widehat{\epsilon}$ is possible. In the first case both events would be equal (remember that $\vartheta(N)$ and $\vartheta'(N)$ are both not $\epsilon$, thus $N$ is the only object net not mapped to $\epsilon$ by $\vartheta$, resp. $\vartheta'$, which implies that the events are equal). The second case is not possible as was already shown above. Thus $\widehat{t} \neq \widehat{t'}$ holds and only two possible cases are left: Either both events are synchronous events or one is a synchronous the other an object-autonomous event. In the first case $\widehat{l(t)}(N) = \widehat{l(t')}(N)$ follows from $\widehat{l(t)}(N) = l_N(\vartheta(N))$, $\widehat{l(t')}(N) = l_N(\vartheta'(N))$ and $\vartheta(N) = \vartheta(N')$, but this is not possible in a *strongly deterministic* GSM. In the second case let w.l.o.g. $\widehat{t}[\vartheta] \in \Theta_l$ and $\widehat{t'}[\vartheta'] = \widehat{\epsilon}[\vartheta'] \in \Theta_o$. But since $\vartheta(N)$ and $\vartheta'(N)$ are both not $\epsilon$ but transitions of $N$, this would mean that $\vartheta(N)$ fires synchronously with $\widehat{t}$ and $\vartheta'(N)$ fires object-autonomously (i.e. is not labelled with a channel). Since $\vartheta(N) = \vartheta'(N)$, this is not possible. Thus like above (in the case $p \in \widehat{P}$) where $\widehat{t} = \widehat{t'}$ was not possible, here (in the case $p \in P_N$) $\vartheta(N) = \vartheta'(N)$ is not possible.[1]

The case $|\mathbf{pre}_{\mathrm{RN}}(p)| > 1$ is again analogously and from this $|\mathbf{pre}_{\mathrm{RN}}(p)| = |\mathbf{post}_{\mathrm{RN}}(p)| = 1$ follows for all $p \in P(\mathrm{RN}(G))$. Thus $\mathrm{RN}(G)$ is a T-net. □

The above proof technique is also used in the context of conflict-free GSM, see Section 4.6.

Since in a strongly deterministic GSM $G$ the number of events is bounded by a polynomial in the size of $G$ (see Lemma 4.15), the reference net of $G$ can be constructed in polynomial time. Since furthermore reachability can be decided in polynomial time for a T-net (see 2.41), we have the following corollary to Theorem 4.25:

**Corollary 4.26.** *The reachability problem for strongly deterministic ttGSMs is solvable in polynomial time.*

While for a strongly deterministic ttGSM $G$ the reference net $\mathrm{RN}(G)$ is a T-net, this is in general not the case for deterministic ttGSMs and general ttGSMs. Figure 4.4 shows an example. The net system on the left is clearly a deterministic ttGSM, but its reference net on the right is not a T-net, since the place $p'$ has two input (and two output) arcs.

[1] Note that $\vartheta(N) = \vartheta'(N)$ is thus not possible and that in this part of the proof the characteristic of a T-net was not used. For this reason, this part is the same as in Theorem 4.35.

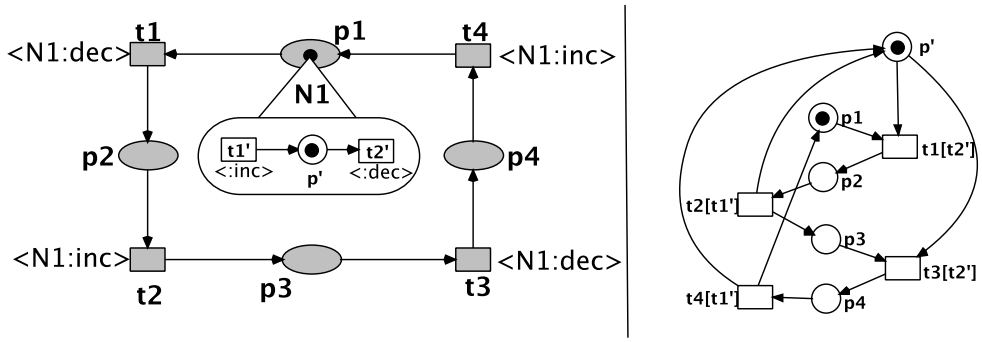


Figure 4.4: A ttGSM $G$ (left) with its reference net $\text{RN}(G)$ (right).

So, not only the size of $\text{RN}(G)$ might become huge in the case of a deterministic or general ttGSM $G$, also the structure of $\text{RN}(G)$ might become more intricate. This was actually the very reason why deterministic and strongly deterministic GSMs were introduced in [HKB11b].

Turning again to ppGSMSs, we note that even if $\text{RN}(G)$ is not a P-net for any of the ppGSM formalisms (see Lemma 4.22 and the discussion following it), the reference net is always 1-conservative and thus reachability can be solved in polynomial space (cf. [JLL77] and Section 2.5). One of the reasons might be that in a ppGSM $G$ only one object net is present and thus the number of events is rather limited independently of $G$ being strongly deterministic, deterministic or a general ppGSM.

To pinpoint the exact complexity of the reachability problem for deterministic ttGSMs is currently an open problem. The more sophisticated structure of $\text{RN}(G)$ in the case of deterministic and general ttGSMs, the increasing number of events and the potential presence of several object nets, are the reasons why ttGSMs are not as well understood yet as ppGSMs.

### ptGSMs.

We now turn our attention to ptGSMs and tpGSMs, i.e. to GSMs which employ a mixture of P- and T-nets. While in the cases of ttGSM and ppGSM above we got some positive results in way of algorithms, for ptGSMs and tpGSMs we up to now only have negative results, i.e. hardness results, or no results at all. For ptGSMs we already gave a proof in [HKB11b] that reachability is NP-hard if the GSM is deterministic (a result that carries over to general ptGSMs), but this proof was faulty. We give a correct proof here. Furthermore, we strengthen (or rather worsen) this result to PSpace-hardness and we show that for strongly deterministic ptGSMs the reachability problem is NP-hard. For tpGSMs we conjecture that the problem is solvable in polynomial time and give some intuition why this might be so.

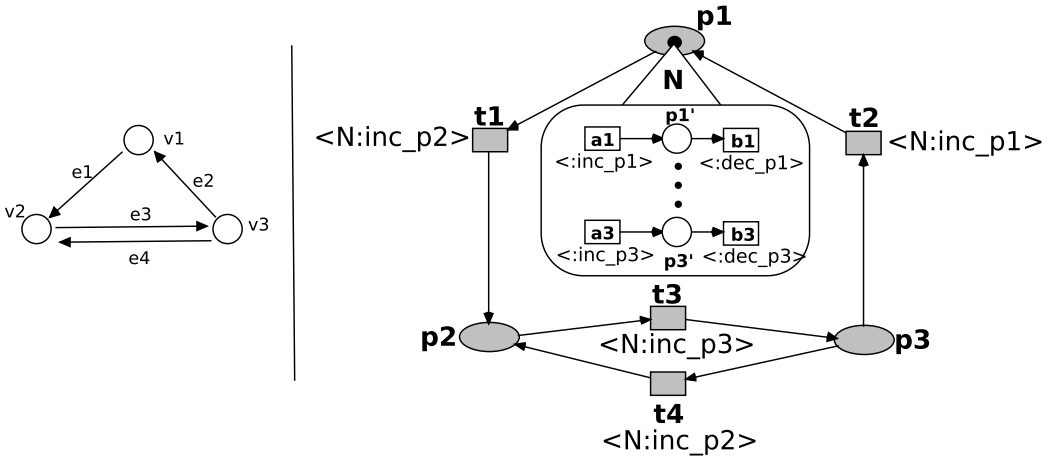


Figure 4.5: Example of the construction in the proof of Theorem 4.27

The following Theorem proves that the reachability problem is NP-hard for deterministic and thus also for general ptGSMs. This result is strengthened in Theorem 4.29 to PSPACE-hardness and is also a special case of Theorem 4.28 showing NP-hardness of the reachability problem for strongly deterministic pt-GSMs. We give the proof here anyway, because it shows a different construction then in the two other proofs.

**Theorem 4.27.** *Let $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a deterministic ptGSM Then the reachability problem is NP-hard.*

*Proof.* We give a reduction from the *Hamilton Circuit* problem for directed graphs (cf. [GJ79]).

Let $DG = (V, E)$ be a directed graph and thus an instance of the *Hamilton Circuit* problem. We assume $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. The transformation is illustrated in Figure 4.5. For each node $v_i \in V$ the system net has a place $p_i$. For each directed edge $e_i = (v_j, v_k)$ the system net has a transition $t_i$ and two arcs $(p_j, t_i)$ and $(t_i, p_k)$. The transition is moreover labelled by $\widehat{l}(t_i)(N) = inc_{p_k}$, where $N \in \mathcal{N}$ will be the sole object net. Each place of the system net is typed with $N$.

For each node $v_i \in V$ the object net $N$ has a place $p'_i$, two transitions $a_i$ and $b_i$, and the arcs $(a_i, p'_i)$ and $(p'_i, b_i)$. The transitions are furthermore labelled by $l_N(a_i) = inc_{p_i}$ and $l_N(b_i) = dec_{p_i}$. Note that the channels $dec_{p_i}$ are not used in the system net and thus the transitions $b_i$ are never enabled.

The initial marking is given by $\mu_0 := p_1[\mathbf{0}]$. The marking tested for reachability is given by $\mu_e := p_1[p'_1 + p'_2 + \ldots + p'_n]$.

The construction is clearly possible in polynomial time in the size of $DG$. Furthermore, the constructed net system is a deterministic ptGSM, which we will denote by $G$.

Now we have to prove that the construction works, i.e. that $DG$ has a Hamilton circuit iff $\mu_e$ is reachable in $G$ from $\mu_0$.

The idea is that the object net $N$ "travels" through the system net and whenever it visits a place $p_i$ – which corresponds to a node $v_i$ in the given graph – it remembers this by placing a black token onto $p'_i$.

Formally, let $C = \langle v_{i_0}, v_{i_1}, \ldots, v_{i_n} \rangle$ be a Hamilton circuit in $DG$. We may assume that $v_{i_0} = v_{i_n} = v_1$, because $C$ is a circuit and it does not matter here at which node it starts.

Let $e_{j_1}, e_{j_2}, \ldots, e_{j_n}$ be the edges such that $e_{j_k} = (v_{i_{k-1}}, v_{i_k})$ for $k \in \{1, \ldots, n\}$.

In $G$ it is now possible to fire the evens $t_{j_1}[a_{j_1}], t_{j_2}[a_{j_2}], \ldots, t_{j_n}[a_{j_n}]$ one after the other, because the only prerequisite for firing is that the object net resides on the place in the preset of the system net transition, which is guaranteed, since $C$ is a circuit starting at $v_1$. Since each node $v_i \in V$ is visited exactly once in $C$, each channel $inc_{p_i}$ is used exactly once and thus each place $p'_i$ in the object net $N$ is marked with exactly one black token. Thus $\mu_e$ is reached.

Conversely, suppose $\mu_e$ is reached from $\mu_0$ in $G$. In $\mu_e$ there are $n$ black tokens in the object net $N$, each on one of its $n$ places. Since there are only synchronous events and each of these events increases the number of black tokens of exactly one place in the object net and since no black token can be removed from a place in the object net, the firing sequence must have length $n$ and thus $n$ events have fired to reach $\mu_e$ from $\mu_0$.

No two events can use the same channel $inc_{p_i}$, because then two black tokens would reside on $p'_i$ in $N$. But if all $n$ events use pairwise different channels all places visited in $\widehat{N}$ differ and since there are only $n$, all $n$ places have been visited by $N$ exactly once.

Let $\langle p_{i_0}, p_{i_1}, \ldots, p_{i_n} \rangle$ be the ordered sequence of places $N$ visits in $\widehat{N}$ with $p_{i_0} = p_{i_n} = p_1$. Then a Hamilton circuit in $DG$ is given by $\langle v_{i_0}, v_{i_1}, \ldots, v_{i_n} \rangle$.

We thus have a polynomial time many-one reduction from the NP-complete *Hamilton Circuit* problem and conclude that the reachability problem is NP-hard for deterministic ptGSMs. □

Note that the GSM constructed above only uses one object net and no black tokens on the system net level. This is actually true for all (connected) ptGSMs (see 4.19). Thus, while for the system net or the object net treated in isolation the reachability problem can be decided in polynomial time, the interaction of these in a ptGSM already lifts the problem to be NP-hard.

In Theorem 4.29 below we show that the problem is even PSpace-hard. Its exact complexity is currently unknown, but we suspect it to be in PSpace. Before that we now show that the problem is also NP-hard for strongly deterministic ptGSMs, which is also an alternative proof of the statement above, since from this result NP-hardness for deterministic and general ptGSMs also follows.

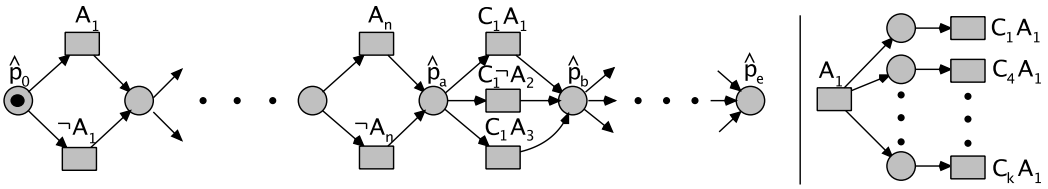


Figure 4.6: System net (left) and object net (right) of the ptGSM from Theorem 4.28.

**Theorem 4.28.** *It is* NP*-hard to decide the reachability problem for strongly deterministic ptGSMs.*

*Proof.* We give a reduction from 3-SAT. The ptGSM we construct is outlined in Figure 4.6, where only the transitions' channels are given. Assume we are given an instance of 3-SAT, i.e. a formula $F$ with $n$ variables $A_1, \ldots, A_n$ and $m$ clauses $C_1, \ldots, C_m$, where each clause consists of exactly three literals.

The idea is that the object net moves through the system net and with its first $n$ steps sets each variable to true or false. The next $m$ moves check if in each clause one literal is true with this setting. All $m$ moves are only possible if the setting is a satisfying assignment.

In more detail: For each variable $A$ a pair of transitions exists in the system net (see Figure 4.6), one for the positive literal $A$ and one for the negative literal $\neg A$. The transitions use the channels $A$ and $\neg A$ respectively. To describe the object net $N$ assume that $L$ is a literal (e.g. $A_1$ or $\neg A_n$) that appears in the clauses $C_{i_1}, \ldots, C_{i_k}$. Then $N$ has one transition $t_L$ using channel $L$ and with empty preset and $k$ outgoing arcs. Each outgoing arc is connected with a place which is connected to a transition using a channel $C_{i_j}L$ corresponding to the clause. In Figure 4.6 (right side) this is only depicted for the positive literal $A_1$. Note that such a construct exists for each literal and that all used channels differ.

Further system net transitions exists to encode the clauses. Let $L_{i,1}, L_{i,2}, L_{i,3}$ be the literals that occur in clause $C_i$. Then three transitions exists in the system net using the channels $C_iL_{i,1}$ to $C_iL_{i,3}$ as depicted in Figure 4.6 for the clause $C_1$ (assumed to be $A_1 \vee \neg A_2 \vee A_3$).

The initial marking is $\mu_0 := \widehat{p}_0[\mathbf{0}]$.

It is easy to see that the object net can reach the place $\widehat{p}_e$ (to the right of the system net) iff the given formula is satisfiable. Unfortunately, this only shows that the marking $\mu_e := \widehat{p}_e[\mathbf{0}]$ can be *covered*.[2]

To get rid of the tokens in the object net a more sophisticated gadget needs to be constructed. The gadget is depicted in Figure 4.7. It replaces the part between the places $\widehat{p}_a$ and $\widehat{p}_b$ in Figure 4.6 and an identical structure only using

[2]Thus the coverability problem is also NP-hard for ptGSMs.

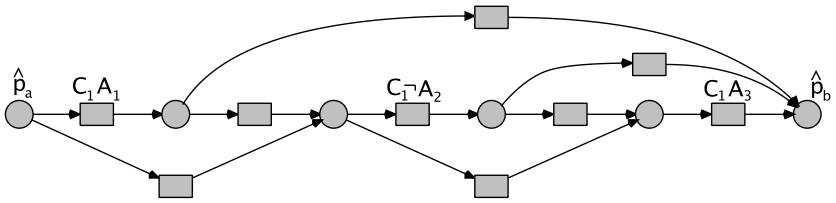


Figure 4.7: Modification of the system net of Figure 4.6.

different channels needs to be added for the other clauses. The gadget has the following property: At least one of the transitions with channel inscriptions needs to fire to reach $\widehat{p}_b$ and apart from the empty set every subset of these transitions is possible. (The unlabelled transitions fire system-autonomously.)

With this gadget it is possible to fully rid the object net of all tokens. If for example the transition with channel $A_1$ has fired in the object net, then in each clause $C_i$ in which $A_1$ appears the system and object net transitions using channel $C_iA_1$ may fire.

Thus the marking $\mu_e = \widehat{p}_e[\mathbf{0}]$ is reachable from $\mu_0 = \widehat{p}_0[\mathbf{0}]$ in the constructed net iff $F$ is satisfiable. Since the construction is possible in polynomial time and the constructed net system is a strongly deterministic ptGSM, we conclude that the reachability problem is NP-hard for this net class. □

Strengthening the result of Theorem 4.27, we prove below by a slight modification of the construction in Lemma 4.20 that reachability is even PSPACE-hard for deterministic and general ptGSMs.

**Theorem 4.29.** *The reachability problem for deterministic and general ptGSMs is* PSPACE*-hard.*

*Proof.* We construct a ptGSM from a given LBA as in the proof of Lemma 4.20. The problem is that each place of the object net constructed there might have many incoming and many outgoing arcs, while here exactly one incoming and exactly one outgoing arc should exist. To circumvent this our new object net has the same set of places but for each place $p$ it has one transition $t_p^+$ with an arc to $p$ and one transition $t_p^-$ with an arc from $p$. The transitions use the channels $c_p^+$ and $c_p^+$ respectively.

The transitions in the system net are replaced by two transitions and one place as follows. If before we had the transition $\widehat{t}_{r,j}$ and the two arcs $(\widehat{p}_{i,j}, \widehat{t}_{r,j})$ and $(\widehat{t}_{r,j}, \widehat{p}_{i+1,j+1})$ as depicted to the left of Figure 4.3, then we replace these with two transitions $\widehat{t}^-_{r,j}, \widehat{t}^+_{r,j}$ and the arcs $(\widehat{p}_{i,j}, \widehat{t}^-_{r,j}), (\widehat{t}^-_{r,j}, \widehat{t}^+_{r,j}), (\widehat{t}^+_{r,j}, \widehat{p}_{i+1,j+1})$. The channels used by the two new transitions depend on the LBA's transition $k_r$. If $k_r = (q_i, A_b, R, q_{i+1}, A_1)$ (again, as depicted in Figure 4.3), then the token denoting that the cell $c_j$ is marked with $A_b$ has to be removed and a token

Table 4.1: The results obtained so far for P- and T-net like GSMs.

| | strongly deterministic | deterministic | general |
|---|---|---|---|
| ttGSM | P | ? | ? |
| ppGSM | PSPACE-complete | PSPACE-complete | PSPACE-complete |
| ptGSM | NP-hard | PSPACE-hard | PSPACE-hard |
| tpGSM | ? | ? | ? |

denoting that the cell $c_j$ is marked with $A_1$ has to be added, i.e. the transition $\widehat{t}^-_{r,j}$ uses channel $c^-_{p_{b,j}}$ and the transition $\widehat{t}^-_{r,j}$ the channel $c^+_{p_{1,j}}$.

The construction is still possible in polynomial time, but the constructed net system is now a ptGSM. Thus the theorem follows. □

### tpGSMs.

We now turn to tpGSMs, which unlike ptGSMs above (see Lemma 4.19) may employ more than one object net.

At first glance this thus seems to be a complicated case, too. However, the structure of the system net is heavily limited by being a GSM *and* a T-net. In this setting each object net $N$ resides on a circle $C_N$ in the system net. While these circles may have transitions in common and while thus the object nets may interact with each other, this interaction is rather limited as are the reachable system net places for each object net.

For this reason, we suspect that the reachability problem for tpGSMs is solvable in polynomial time. So far we have been unable to prove this conjecture. If true, the results obtained would imply that the restriction of the system net to a T-net is far more severe than the restriction of the system net to a P-net. This would also be in line with the results presented here for tt- and ppGSMs, where reachability is solvable in polynomial time for a strongly deterministic ttGSM, but requires polynomial space for a ppGSM.

### Summary.

Table 4.1 summarizes the results obtained so far concerning the complexity of the reachability problem for the the different variants of generalised state machines introduced in this section. (Note that the problem is decidable for all of them due to Theorem 4.9.) While GSMs with the restrictions applied here will probably not be very useful in a modelling context, the theoretical results obtained might help in the analysis of less restricted models. For example the technique using the reference net exhibit in the proofs of Theorems 4.22 and 4.25 is also used in the proof of a similar statement for conflict-free GSMs (see Theorem 4.35).

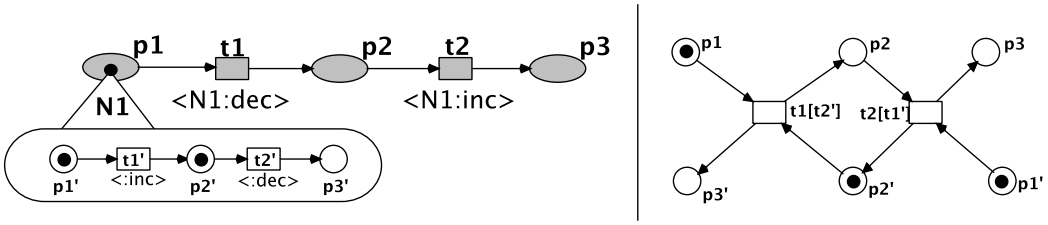


Figure 4.8: A strongly deterministic and acyclic GSM (left), with a non-acyclic reference net (right).

## 4.5 Acyclic Generalised State Machines

As was the case for P- and T-nets, the definition of acyclic p/t nets (Definition 2.44) can also be easily carried over to generalised state machines. A GSM is called acyclic if the system net and all object nets are acyclic, i.e. no cycles exist in the structure of the nets.

**Definition 4.30 (Acyclic GSMs).** *Let $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a GSM. $G$ is an* acyclic GSM *or* acGSM *for short, if $\widehat{N}$ and all $N \in \mathcal{N}$ are acyclic nets, i.e. if there is no $x \in \widehat{P} \cup \widehat{T}$ with $(x, x) \in \widehat{F}^+$ and there is no $x \in P_N \cup T_N$ with $(x, x) \in F_N^+$ for no $N \in \mathcal{N}$.*

Since the reachability problem for acyclic p/t nets is NP-hard (see Theorem 2.45) and since acyclic p/t nets are a special case of acyclic GSMs, this hardness result carries over to acyclic GSMs.

**Theorem 4.31.** *The reachability problem for strongly deterministic, deterministic, and general acGSMs is* NP*-hard.*

The positive result that the reachability problem for acyclic p/t nets is also in NP does not carry over so easily to acyclic GSMs, at least not by usage of the reference net, because, unfortunately, the absence of cycles does not carry over to the reference net not even for strongly deterministic acGSMs. Figure 4.8 shows a GSM that is acyclic and strongly deterministic, but whose reference net is not acyclic.

Even if one does demand safeness (cf. Definition 5.8) in addition to the absence of cycles, the reference net must not necessarily be acyclic by the same example (cf. Section 5.4).

If only autonomous events are allowed the reachability problem can be easily reduced to the same problem for acyclic p/t nets.

**Theorem 4.32.** *The reachability problem for acyclic GSMs with no synchronous events is* NP*-complete.*

*Proof.* NP-hardness follows as before, since an acyclic p/t net $N$ can be used as the sole object net of a GSM $G$ that resides on the sole system net place $\widehat{p}$. The question if a marking $m$ is reachable in $N$ from $m_0$ is then reduced to the question if $\widehat{p}[m]$ is reachable in $G$ from $\widehat{p}[m_0]$.

An NP-algorithm that solves the reachability problem given a acyclic GSM $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ with no synchronous events and a marking $\mu$ works as follows: At first, the reachability problem for the acyclic net system $(\widehat{N}, \Pi^1(\mu_0))$ and marking $\Pi^1(\mu)$ is solved, i.e. for the system net of $G$ where all net tokens are interpreted as black tokens. If this is possible, the net tokens can reach their destinations. For this it is important that in a acyclic generalised state machine there is only one object net of each type and that it may reach each place only once. Thus there is no choice which object net should reach which place.

Afterwards, the reachability problem is solved in each object net alone, i.e. the problem if $\Pi^2_N(\mu)$ is reachable from $\Pi^2_N(\mu_0)$ in $N$ for each $N \in \mathcal{N}$.

This solves the reachability problem by running $|\mathcal{N}| + 1$ instances of the NP-algorithm for the reachability problem for acyclic p/t net systems. □

Alternatively to the proof of Theorem 4.32, one could also observe that the reference net of a GSM as described in the statement of the theorem is an acyclic p/t net system that consists of the system net and all object nets as separated components.

In the general case, the possible interactions of the system net with the object nets lead to complications. Each system net transition that has at least one object net $N \neq N_\bullet$ in its preset is only able to fire at most once, because the object net of type $N$ may reside on this place only once in an acyclic net and there is only one net of this type due to the properties of GSMs. One could thus attempt to guess the firing sequence of system net transitions. However, this does neither hold for system net transition which have only places typed with $N_\bullet$ in their preset, nor for the object nets. In particular, it might be necessary to fire object autonomous events of an object net $N$, before $N$ participates in a synchronous event, and while guessing might again help here, it might be that the sequence becomes too long.

While this observation helps to determine the complexity of the reachability problem for further subclasses of acyclic GSMs – for example the case of acyclic GSMs with only synchronous events and where no system net place is typed with $N_\bullet$ can be solved in NP with the approach presented above, i.e. by guessing the sequence of system net transitions and if necessary the object net transitions that fire synchronously –, the complexity of the reachability problem for general acyclic GSMs is still unknown. To solve the problem, a better understanding of how the black tokens mingle with the object nets is necessary. Intuitively, only the creation of black tokens is problematic, because each token can only be in the preset of a transition once. Then again, it seems that the creation of

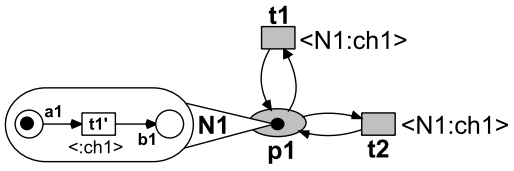


Figure 4.9: An Eos where $t_1$ and $t_2$ are in conflict

tokens is more or less independent from the object nets that have "moved on" and can not return, since no circles are allowed in the system net.

## 4.6 Conflict-free Generalised State Machines

The concept of conflict-freedom can not be so easily carried over to GSMs as the concepts before that lead to ppGSMs, ttGSMs, ptGSMs, tpGSMs, and acyclic GSMs. We will illustrate the problem by discussing conflict-freedom for Eos first.

At first, one might be tempted to define conflict-free Eos similar to conflict-free p/t nets and simply demand that $p^\bullet \subseteq {}^\bullet p$ holds for each place $p$ with $|p^\bullet| > 1$ regardless of $p$ being a system or object net's place. Unfortunately, this attempt does not work due to the capability of the channels. Figure 4.9 illustrates the problem. The net is clearly conflict-free in the sense of the above definition. Two events, namely $t_1[t_1']$ and $t_2[t_1']$ are enabled, but after firing one of them the new marking is $p_1[b_1]$ and the other event is not enabled anymore (since the object net transition $t_1'$ is not enabled anymore).

One might try to include the channels in the definition (indeed the place $a_1$ in the above example can be thought of having *two* transitions in it's post-set), but this also does not work. The left side of Figure 4.10 shows another example, in which the places $p_1$ and $p_2$ are both typed with the same object net $N_1$. This time the two enabled events $t_1[t_1']$ and $t_2[t_2']$ use different channels, but firing one *might* deactivated the other - depending on the mode used during firing and in particular the distribution of the object net's tokens after firing.

Imagine the event $t_1[t_1']$ takes place. Afterwards, there will be one net token on places $p_1$ and $p_2$ each. The black token on place $a_2$ and $b_1$ can be distributed *arbitrarily* among these two net tokens depending on the mode used during firing. Thus, the successor marking might be $p_1[b_1] + p_2[a_2]$ in which the event $t_2[t_2']$ is not enabled anymore.

This example already shows that it is the firing rule itself that makes defining conflict-free Eos problematic. On the right side of Figure 4.10 is another example. Here not even channels are present, but if $t_1$ fires system-autonomously the object net's tokens may be distributed among the object net's on places $p_2$

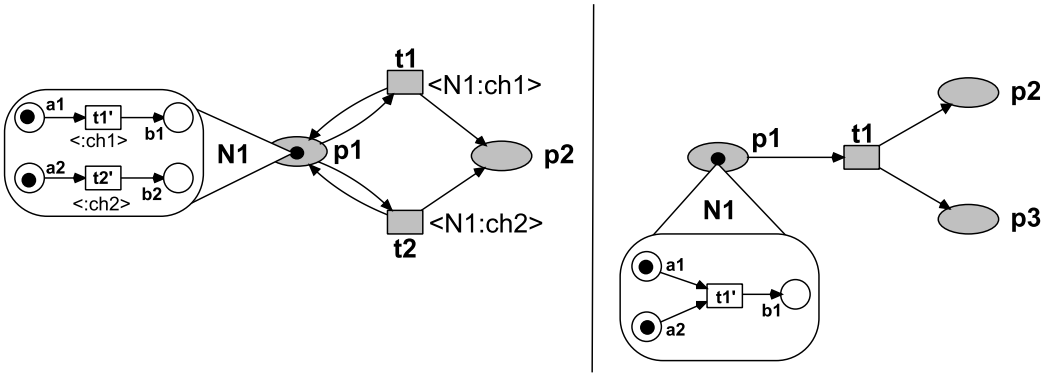


Figure 4.10: Two Eos where conflicts depend on the firing mode

and $p_3$ in a way that $t_1'$ is not enabled anymore (e.g. in the successor marking $p_2[a_1] + p_3[a_2]$).

To conclude, the example in Figure 4.9 shows that the structural definition from p/t nets can not be so easily carried over to Eos due to the channels. Furthermore, the examples in Figure 4.10 show that the firing rule for Eos itself is inherently problematic for the definition of conflict-freedom, whenever a fork is present, e.g. a transition with two places with the same typing in its post-set.

Turning to GSM we first note that the problem depicted in Figure 4.10 can not arise, because a fork of this kind is not possible. The problem depicted in Figure 4.9, however, can arise and is dealt with in item 2 in the definition of conflict-free GSMs below.

**Definition 4.33 (Conflict-Free GSMs).** *A GSM* $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *is* conflict-free *if*

1. $\forall N \in \mathcal{N} \cup \{\widehat{N}\}\ \forall p \in \widehat{P} \cup P_{\mathcal{N}} : |p^\bullet| > 1 \Rightarrow p^\bullet \subseteq {}^\bullet p$
2. $\forall N \in \mathcal{N}\ \forall p \in P_N : (\exists t \in p^\bullet\ \exists \widehat{t_1}, \widehat{t_2} \in \widehat{T}\ \exists \widehat{p} \in \widehat{P}\ \exists c \in C : \widehat{t_1} \neq \widehat{t_2} \wedge \widehat{p} \in {}^\bullet\widehat{t_1} \cap {}^\bullet\widehat{t_2} \wedge d(\widehat{p}) = N \wedge l_N(t) = \widehat{l}(\widehat{t_1})(N) = \widehat{l}(\widehat{t_2})(N) = c) \Rightarrow p \in t^\bullet$

*holds. We also say that OS is a* cfGSM.

The first part of the definition simply carries over the definition for p/t nets to GSMs. Viewed as a p/t net each object net and the system net are conflict-free.

The second part is necessary due to the problem of the example in Figure 4.9. When a place $p$ of an object net $N$ in the preset of a transition $t$ exists such that there is a system net place $\widehat{p}$ typed with $N$ and two system net transitions $\widehat{t_1}$ and $\widehat{t_2}$ which both have $\widehat{p}$ in their presets and if all these transitions use the same channel then we have exactly the situation in Figure 4.9 and thus we

demand that $p$ is also in the postset of $t$ (in the example this would force an arc from $t_1'$ to $a_1$). Note that the second item holds unconditionally for strongly deterministic GSMs, because no two system net transitions may use the same channel and thus the premise of the implication is never fulfilled.

Lower bounds concerning the reachability problem can again be carried over from p/t nets.

**Theorem 4.34.** *The reachability problem for strongly deterministic, deterministic, and general conflict-free GSMs is* NP*-hard.*

*Proof.* Let $N$ be a conflict-free p/t net system. $N$ can be used as the sole object net present in a conflict-free GSM $G$ that has only one system net place $\widehat{p}$ on which $N$ resides.

The question if a marking $m$ is reachable from $m_0$ in $N$ is reduced to the question if $\widehat{p}[m]$ is reachable from $\widehat{p}[m_0]$ in $G$. □

We now prove that known complexity results for conflict-free p/t nets can be carried over to specific GSMs. The technique used in the following proof is also used in the proofs of Theorems 4.22 and 4.25.

**Theorem 4.35.** *Let $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a strongly deterministic and conflict-free GSM. The p/t net* $\textsc{Rn}(G)$ *is a conflict-free p/t net.*

*Proof.* Let $p \in P(\textsc{Rn}(G))$ and let $|p^\bullet| > 1$. Let $\widehat{t}[\vartheta] \in p^\bullet$. We want to show $\widehat{t}[\vartheta] \in {}^\bullet p$, which implies $p^\bullet \subseteq {}^\bullet p$. Because of $|p^\bullet| > 1$ there exists an event $\widehat{t'}[\vartheta']$ different from $\widehat{t}[\vartheta]$ with $\widehat{t'}[\vartheta'] \in p^\bullet$. From the definition of $\mathbf{pre}^{\textsc{Rn}}$ in Definition 4.7 it follows that

$$\begin{aligned} & (p \in \mathbf{pre}(\widehat{t}) \vee p \in \mathbf{pre}_N(\vartheta(N)) \text{ for a } N \in \mathcal{N}) \\ \wedge \quad & (p \in \mathbf{pre}(\widehat{t'}) \vee p \in \mathbf{pre}_{N'}(\vartheta'(N')) \text{ for a } N' \in \mathcal{N}) \end{aligned}$$

We now distinguish the two cases $p \in \widehat{P}$ and $p \in P_N$ for a $N \in \mathcal{N}$.

In the first case $p \in \mathbf{pre}(\widehat{t}) \wedge p \in \mathbf{pre}(\widehat{t'})$ follows, which implies $\widehat{t} \neq \widehat{\epsilon} \neq \widehat{t'}$. The events are thus not object-autonomous. With this and from $\widehat{t}[\vartheta] \neq \widehat{t'}[\vartheta']$, $\widehat{t} \neq \widehat{t'}$ follows. Assume otherwise. Then a net $N \in \mathcal{N}$ would have to exist with $\vartheta(N) \neq \vartheta'(N)$. But with $\vartheta(N) = t_1 \neq t_2 = \vartheta'(N)$ synchronisation would require that $l_N(t_1) = \widehat{l}(\widehat{t})(N) = l_N(t_2)$, which is not possible in a deterministic GSM, if $t_1, t_2 \in T_N$. (Note that for $t \in T_N$, $\vartheta(N) = t$ implies $l_N(t) \neq \tau$.) If w.l.o.g. $t_1 \in T_N$ and $t_2 = \epsilon$, then $l_N(t_2) = l_N(\epsilon) = \tau = l_N(t_1)$ would mean that $t_1$ fires object-autonomously in contrast to $\widehat{t'} \neq \widehat{\epsilon}$. With $\widehat{t} \neq \widehat{t'}$ the first item of Definition 4.33 implies $p \in \mathbf{post}(\widehat{t}) \wedge p \in \mathbf{post}(\widehat{t'})$ from which $p \in \mathbf{post}^{\textsc{Rn}}(\widehat{t}[\vartheta])$ (and also $p \in \mathbf{post}^{\textsc{Rn}}(\widehat{t'}[\vartheta'])$) or equivalently $\widehat{t}[\vartheta] \in {}^\bullet p$ follows.

In the second case ($p \in P_N$ for a $N \in \mathcal{N}$) a net $N \in \mathcal{N}$ exists such that $p \in \mathbf{pre}_N(\vartheta(N)) \wedge p \in \mathbf{pre}_N(\vartheta'(N))$ holds.[3] This implies $\vartheta(N) \neq \epsilon \neq \vartheta'(N)$. The events are thus not system-autonomous.

In the case of $\vartheta(N) \neq \vartheta'(N)$, $p \in \mathbf{post}_N(\vartheta(N)) \wedge p \in \mathbf{post}_N(\vartheta'(N))$ follows from the first item of Definition 4.33 again, which immediately implies $p \in \mathbf{post}^{\mathrm{R_N}}(\widehat{t}[\vartheta])$, resp. $\widehat{t}[\vartheta] \in {}^\bullet p$.

In the case of $\vartheta(N) = \vartheta'(N)$, neither $\widehat{t} = \widehat{t'} = \widehat{\epsilon}$, nor $\widehat{t} = \widehat{t'} \neq \widehat{\epsilon}$ is possible. In the first case both events would be equal (remember that $\vartheta(N)$ and $\vartheta'(N)$ are both not $\epsilon$, thus $N$ is the only object net not mapped to $\epsilon$ by $\vartheta$, resp. $\vartheta'$, which implies that the events are equal). The second case is not possible as was already shown above. Thus $\widehat{t} \neq \widehat{t'}$ holds and only two possible cases are left: Either both events are synchronous events or one is a synchronous the other an object-autonomous event. In the first case $\widehat{l(t)}(N) = \widehat{l(t')}(N)$ follows from $\widehat{l(t)}(N) = l_N(\vartheta(N))$, $\widehat{l(t')}(N) = l_N(\vartheta'(N))$ and $\vartheta(N) = \vartheta(N')$, but this is not possible in a *strongly deterministic* GSM. In the second case let w.l.o.g. $\widehat{t}[\vartheta] \in \Theta_l$ and $\widehat{t'}[\vartheta'] = \widehat{\epsilon}[\vartheta'] \in \Theta_o$. But since $\vartheta(N)$ and $\vartheta'(N)$ are both not $\epsilon$ (but transitions of $N$), this would mean that $\vartheta(N)$ fires synchronously with $\widehat{t}$ and $\vartheta'(N)$ fires object-autonomously (i.e. is not labelled with a channel). Since $\vartheta(N) = \vartheta'(N)$, this is not possible. Thus like above (in the case $p \in \widehat{P}$) where $\widehat{t} = \widehat{t'}$ was not possible, here (in the case $p \in P_N$) $\vartheta(N) = \vartheta'(N)$ is not possible.

All cases treated, $\widehat{t}[\vartheta] \in {}^\bullet p$ follows, which implies $p^\bullet \subseteq {}^\bullet p$ and thus $\mathrm{R_N}(G)$ is conflict-free. □

Since in a strongly deterministic or deterministic GSM the number of events is bounded by a polynomial (see Lemma 4.15), the reference net can be constructed in polynomial time. In [HR88] Howell and Rosier have proven that for conflict-free p/t nets the reachability problem is NP-complete (see also Section 2.5). This implies the following corollary:

**Corollary 4.36.** *The reachability problem for strongly deterministic, conflict-free GSMs is* NP*-complete.*

The NP-completeness-result of the reachability problem for conflict-free p/t nets thus carries over to strongly deterministic and conflict-free GSMs. In Chapter 5.4 the results for conflict-free and safe p/t nets (see [HR89] and Section 2.5) are carried over to strongly deterministic, conflict-free, and *safe* GSMs (see Theorem 5.35). Both problems are solvable in polynomial time. The case for GSMs which are not strongly deterministic is not treated here, the complexity of it is - apart from the decidability result from Theorem 4.9 above and the NP-hardness bound inherit from the strongly deterministic case - currently unknown. The problem is, that the set of events $\Theta$ of a GSM $G$

[3] Note that the object nets can not be different, since $p \in P_N$ holds.

might become large and that it might not be possible to construct the reference net $\text{Rn}(G)$ within the given time bound. Then again, it might be enough to construct parts of the reference net on the fly.

For Eos it seems impossible to introduce conflict-freedom due to the firing rule. One can introduce persistent Eos, however, which dynamically rule out conflicts. The formalism is Turing-complete, though (see Chapter 5).

## 4.7 Free-Choice Generalised State Machines

As was possible for acyclic GSMs, the definition of free-choice p/t net systems can be carried over to GSMs. Nonetheless, the intuition captured by this definition for p/t net systems, i.e. that "choices are free", does, like the notion of conflict-freeness, not so easily carry over to GSMs due to the labelling. Here we are satisfied with merely carrying over the structural restrictions to GSMs, because the resulting free-choice GSMs nicely illustrate how complicated the resulting formalisms can become.

**Definition 4.37 (Free-choice GSMs).** *Let $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a GSM. $G$ is a* free-choice GSM *or* fcGSM *for short if $\widehat{N}$ and all $N \in \mathcal{N}$ are free-choice nets, i.e. if for every two places $\widehat{p}_1, \widehat{p}_2 \in \widehat{P}$ and two transitions $\widehat{t}_1, \widehat{t}_2 \in \widehat{T}$ the arcs $(\widehat{p}_1, \widehat{t}_1), (\widehat{p}_1, \widehat{t}_2), (\widehat{p}_2, \widehat{t}_1) \in \widehat{F}$ imply $(\widehat{p}_2, \widehat{t}_2) \in \widehat{F}$ and if for every $N \in \mathcal{N}$ and for every two places $p_1, p_2 \in P_N$ and two transitions $t_1, t_2 \in T_N$ the arcs $(p_1, t_1), (p_1, t_2), (p_2, t_1) \in F_N$ imply $(p_2, t_2) \in F_N$.*

Free-choice GSMs are thus defined in a similar fashion as acyclic GSMs, i.e. the notion known from p/t nets is simply carried over to GSMs by demanding that the system net and each object net is a free-choice net. As was the case with acyclic GSMs, the reference net does, unfortunately, not inherit the property of the GSM: The reference net of a free-choice GSM is not necessarily a free-choice net, even if the GSM is strongly deterministic. Figure 4.11 shows an example. While in the reference net the arcs $(p_2, t), (p_2, t'[t_1, t_2])$, and $(q_1, t'[t_1, t_2])$ are present, the arc $(q_1, t)$ is missing in violation with the requirements for a free-choice p/t net.

While for the structural restrictions introduced so far, it was in many cases possible to either find an algorithm for the reachability problem with an acceptable performance or to only find lower bounds for this problem which are not too bad, the situation for free-choice GSMs is far worse. The reachability problem for free-Choice GSMs is ExpSpace-hard. This can be shown by reducing the reachability problem for p/t nets to it.

**Theorem 4.38.** *The reachability problem for deterministic free-choice GSMs is* ExpSpace*-hard.*

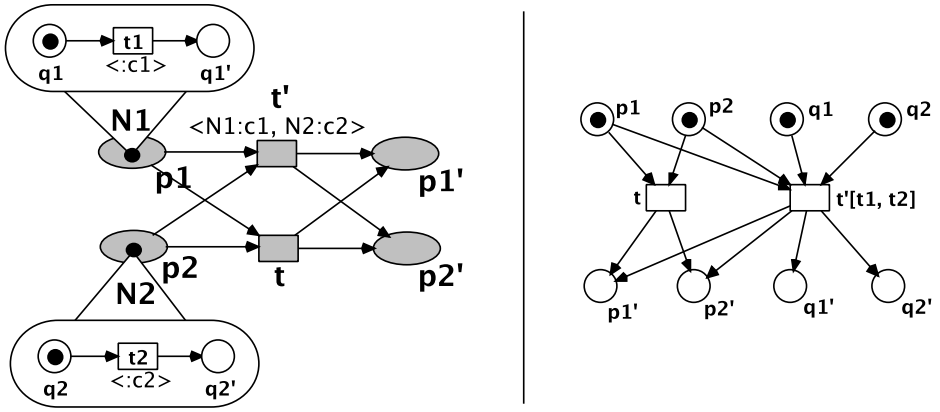


Figure 4.11: An strongly deterministic free-choice GSM (left), with a non-free-choice reference net (right).

*Proof.* Given an instance $(N, m_0, m)$ of the reachability problem for p/t net systems, where the question is, if in the net $N = (P, T, F)$ the marking $m$ is reachable from the initial marking $m_0$, a deterministic GSM $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ and marking $\mu$ can be constructed such that $\mu$ is reachable from $\mu_0$ in $G$ if and only if $m_0 \xrightarrow{*} m$ in $N$.

$G$ is outlined in Figure 4.12. More specific, for each $p_i \in P$ a object net $N_i$ with place $p_i$ and transitions $t_i$ and $t_i'$ is created where $t_i$ is labelled with $inc_i$ and $t_i'$ with $dec_i$. Arcs go from $t_i$ to $p_i$ and from $p_i$ to $t_i$. The object net $N_i$ initially resides on $\widehat{p}_i$. In Figure 4.12 the object net $N_1$ is depicted.

For each transition $t_i \in T$ a system net transition $\widehat{t}_i$ is created and $n := |P|$ places $\widehat{p}_{1,t_i}, \ldots, \widehat{p}_{n,t_i}$ are created in the postset of $\widehat{t}_i$. Furthermore, in the preset of $\widehat{t}_i$ are the places $\widehat{p}_1, \ldots, \widehat{p}_n$ created before.

A second transition $\widehat{t}_i'$ is created that has the places $\widehat{p}_{1,t_i}, \ldots, \widehat{p}_{n,t_i}$ in its pre- and the places $\widehat{p}_1, \ldots, \widehat{p}_n$ in its postset.

The system net's transitions are labelled in the following way: For each place $p_j \in {}^\bullet t_i$ in $N$ the transition $\widehat{t}_i$ is labelled with the channel $dec_j$ (for $N_j$). For each place $p_k \in t_i^\bullet$ the transition $\widehat{t}_i'$ is labelled with the channel $inc_k$ (for $N_k$).

The places $\widehat{p}_i, \widehat{p}_{i,t_1}, \ldots, \widehat{p}_{i,t_m}$ are typed with $N_i$ where $m := |T|$ is the number of transitions in $N$.

If $p_i$ is marked with $c$ black tokens in a marking $m$ of $N$, the place $p_i$ in the object net $N_i$ is marked with $c$ tokens and $N_i$ resides on the system net place $\widehat{p}_i$. The initial marking $m_0 = \sum_{i=1}^{n} c_i \cdot p_i$ is thus converted to $\mu_0 = \sum_{i=1}^{n} \widehat{p}_i[c_i \cdot p_i]$ and the marking tested for reachability in the same way. A marking of the GSM that in this way corresponds to a marking $m$ of $N$ is denoted by $\mu(m)$.

This completes the construction, which is clearly possible in polynomial time.

Now, in a marking $m$ of $N$ a transition $t_i$ is activated if and only if the transition $\widehat{t}_i$ is activated in $G$ in $\mu(m)$, since by construction $\widehat{t}_i$ uses the channel

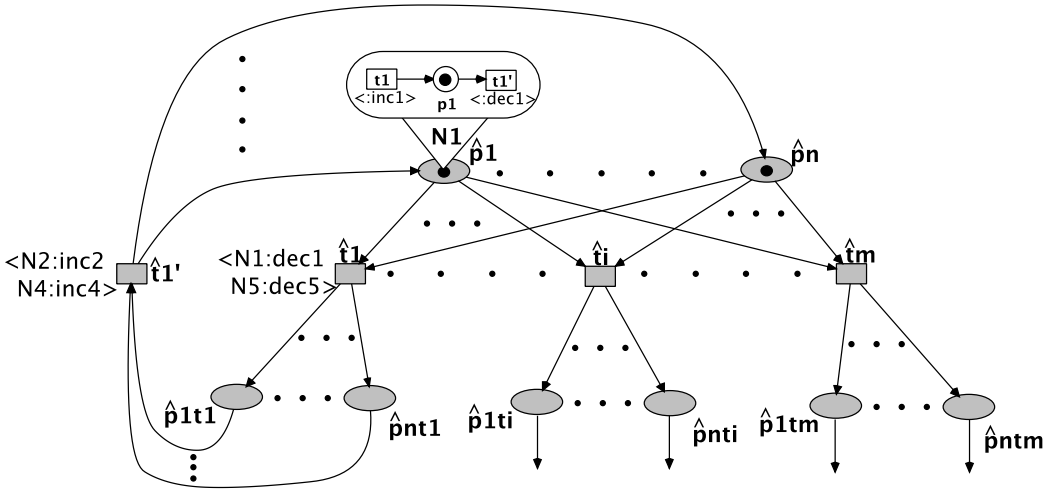


Figure 4.12: Sketch of the construction in the proof of Theorem 4.38

$dec_j$ of $N_j$ and thus decreases the number of tokens on $p_j$ if and only if the place $p_j$ is in the preset of $t_i$. After firing $\widehat{t_i}$ the only activated event is the system net transition $\widehat{t_i'}$ together with the synchronous object net transitions. Firing the event, places the object nets on the places $\widehat{p_1}, \ldots, \widehat{p_n}$ again and since the choice of channels by construction increases the number of tokens in exactly the places in the postset of $t_i$, firing $\widehat{t_i}[\vartheta]$ and $\widehat{t_i'}[\vartheta']$ results in the marking $\mu(m')$, where $m'$ is the successor of $m$ in $N$ after firing $t_i$.

This shows that, if a marking $m$ is reachable in $N$ from $m_0$ by firing a sequence $t_{i_1}, t_{i_2}, \ldots, t_{i_k}$ of transitions, then $\mu(m)$ is reachable from $\mu(m_0) = \mu_0$ by firing the sequence $\widehat{t_{i_1}}[\vartheta_{i_1}], \widehat{t_{i_1}'}[\vartheta_{i_1}'], \widehat{t_{i_2}}[\vartheta_{i_2}], \widehat{t_{i_2}'}[\vartheta_{i_2}'], \ldots, \widehat{t_{i_k}}[\vartheta_{i_k}], \widehat{t_{i_k}'}[\vartheta_{i_k}']$.

If, conversely, a marking $\mu(m)$ is reachable from $\mu(m_0) = \mu_0$ in $G$, then after an event $\widehat{t_i}[\vartheta]$ the only possible event is $\widehat{t_i'}[\vartheta]'$ and these two events corresponds to firing $t_i$ in $N$ thus the sequence of events in $G$ can be transformed into a sequence of transitions in $N$ and thus $m$ is reachable from $m_0$ in $N$. □

While the GSM constructed in the proof of Theorem 4.38 is deterministic, it is not strongly deterministic, since the system net transitions may use the same channel. For example, if two transitions $t_1, t_2$ exists in $N$ that have the same place $p_j$ in their preset, then the transitions $\widehat{t_1}$ and $\widehat{t_2}$ are both labelled with the channel $dec_j$ (for $N_j$). With a slight modification of the construction above it is also possible to construct a strongly deterministic GSM.

**Theorem 4.39.** *The reachability problem for strongly deterministic free-choice GSMs is* ExpSpace*-hard.*

*Proof.* The construction in the proof of Theorem 4.38 needs to be modified only slightly and similarly to the modifications done in Theorem 4.12. In the object net $N_i$ not only one transition is in the preset of $p_i$ but $m := |T|$

transitions $t_{i,1}, \ldots, t_{i,m}$. In the postset of $p_i$ are $m$ transitions $t'_{i,1}, \ldots, t'_{i,m}$, too. The transition $t_{i,j}$ is labelled with channel $inc_{i,j}$ and the transition $t'_{i,j}$ with channel $dec_{i,j}$.

In the system net the labelling is adjusted as follows: If before a system net transition $\widehat{t_j}$ was labelled with channel $dec_i$ (for $N_i$), it is now labelled with channel $dec_{i,j}$, modelling that the number of tokens on place $p_i$ is reduced by the transition $t_j$. Similarly, if a system net transition $\widehat{t'_j}$ was labelled with channel $inc_i$, it is now labelled with channel $inc_{i,j}$. In this way all transitions use different channels.

The proof that the construction is correct is still possible as before and the construction is still possible in polynomial time, which proves the statement. □

The constructions above show that the reachability problem for strongly deterministic free-choice GSMs is ExpSpace-hard. The nets constructed, however, have nice properties if seen as p/t nets: The system net in isolation, i.e. the free-choice p/t net system $(\widehat{N}, \Pi^1(\mu_0))$, is a live, bounded (even 1-safe), and cyclic free-choice p/t net and all object nets are very simple nets. The reachability problem for these nets can be solved in polynomial time (see Theorems 2.41 and 2.49). Their composition here, however, lifts the complexity as far as ExpSpace-hardness.

## 4.8 Summary

Elementary object systems as introduced in Chapter 3 are Turing-complete and thus many important problems are undecidable. To find a formalism which still has at least some of the modelling capabilities of Eos, but which also allows a automatic verification of certain properties, we investigated several structural restrictions of Eos in this chapter.

We presented conservative Eos where each object net type appearing in the preset of a system net transition $\widehat{t}$ also has to appear in the postset of $\widehat{t}$. The test if a object net's marking is $\mathbf{0}$ or not, implicitly possible in an Eos due to the firing rule, is then not possible anymore and, indeed, conservative Eos are not Turing-complete anymore. This follows from the decidability of the termination, the coverability, and the boundedness problem. Unfortunately, liveness and reachability remain undecidable.

We then presented generalised state machines. For each generalised state machine a p/t net can be constructed which has the same behaviour implying the decidability of problems for generalised state machines, which are decidable for p/t nets.

Unfortunately, even for small GSMs the reference net might become huge due to the set of events present in the definition of the reference net. To deal with this problem we introduced deterministic and strongly deterministic GSMs and

Table 4.2: Complexity of the reachability problem for various formalisms.

| | strongly deterministic | deterministic | general |
|---|---|---|---|
| ttGSM | P | ? | ? |
| ppGSM | PSpace-complete | PSpace-complete | PSpace-complete |
| ptGSM | NP-hard | PSpace-hard | PSpace-hard |
| tpGSM | ? | ? | ? |
| acGSM | NP-hard | NP-hard | NP-hard |
| cfGSM | NP-complete | NP-hard | NP-hard |
| fcGSM | ExpSpace-hard | ExpSpace-hard | ExpSpace-hard |
| GSM | ExpSpace-hard | ExpSpace-hard | ExpSpace-hard |
| cEos | undecidable | undecidable | undecidable |
| Eos | undecidable | undecidable | undecidable |

then established several structural restrictions known for p/t nets for GSMs. We introduced pp-, tt-, pt-, and tpGSMs, acyclic, conflict-free, and free-choice GSMs and investigated the reachability problem for each of these formalisms. The problem tends to be complicated even for heavily restricted formalisms. For ppGSMs the problem is PSpace-complete even if the ppGSMs is strongly deterministic. For free-choice GSMs the problem is even ExpSpace-hard. The results obtained in this section are summarised in Table 4.2. The abbreviation cEos is used for conservative Eos.

The construction presented for conservative Eos here and the undecidability result for the liveness problem deduced from it, arose from joint work with Michael Köhler-Bußmeier and is published in [KBH12].

Determinism and strong determinism for GSMs have been introduced in [HKB11a] and were generalised here for Eos. The results for deterministic and strongly deterministic Eos and conservative Eos are presented here for the first time. The results concerning conflict-free GSMs have also been published in [HKB11a]. The results concerning P- and T-nets in the context of GSMs have been published in [HKB12b] and [HKB11b]. These results are published as joint work with Michael Köhler-Bußmeier. The results concerning acyclic GSMs and free-choice GSMs in Sections 4.5 and 4.7 are new and presented here for the first time. A more detailed discussion of acyclic object net systems can be found in the bachelor thesis by Laura Schmelter [Sch12] (in German only), which I supervised.

In retrospect, we applied some long known techniques to solve problems in our new setting and also introduced a new technique.

The idea used in the proof of PSpace-hardness in Lemma 4.20 was used in the late seventies by Jones, Landweber, and Lien [JLL77] and they even state that this technique has already been used by Petri in his dissertation. Another technique, used in the proof of Lemma 4.21, stems from Savitch [Sav70]. A proof technique that we also use in Chapter 5.

The proofs used for ppGSMs (Lemma 4.22), strongly deterministic ttGSMs (Theorem 4.25), and strongly deterministic, conflict-free GSMs (Theorem 4.35) all prove that certain structural restrictions also hold for the reference net. This new "reference net technique" was thus applied successfully three times.

This technique does *not* work for ptGSMs, acyclic GSMs, and free-choice GSMs and does not seem to work for tpGSMs, either.

Thus the technique, while working in some cases, fails to work in some net classes even if these are very restricted.

While some cases are successfully solved, it is interesting to note that all these techniques seem to fail in the case of ptGSMs, tpGSMs, acyclic GSMs, and free-choice GSMs. Despite being quite simple in structure, the possibility to interact seems to render the techniques, successfully applied to the other net classes, useless. This is especially surprising in the case of ptGSMs where only one object net is present. In this regard the high complexity evident in the second and third row of Table 4.2 is also surprising, since both net classes employ only one object net. Recalling the discussion at the end of Section 4.4 concerning tpGSMs, it seems that the restriction of the system net to a T-net results in easier cases than the restriction of the system net to a P-net, despite the fact that more than one object net might be present. While this speculation still needs to be confirmed, it is already certain that the restriction of the system net to a P-net results in hard cases, regardless of the restriction imposed on the object net.

Although some cases are left open and upper bounds are seemingly hard to establish, it is evident from the investigations in this chapter that structural restrictions of object net systems alone are not enough if one strives for a formalism for which the reachability problem can be quickly decided. The case of free-choice GSMs and in particular the example sketched in Figure 4.12 illustrates this best: If the system net is treated as a p/t net where the object nets are abstractly seen as black tokens, it is a live, bounded, and cyclic free-choice p/t net and thus reachability can be solved in polynomial time (Theorem 2.49). The object nets treated as p/t nets are T-nets and thus reachability can be solved in polynomial time, too (Theorem 2.41). The reachability problem, however, is ExpSpace-hard even for these special free-choice GSMs.

In the next chapter, we turn to dynamic restrictions and introduce persistent, unary, and safe Eos. Safe Eos are Eos with a certain restriction on the reachable states that renders the state space finite. In Section 5.3 we prove the surprising result that for safe Eos without any structural restriction not only reachability and liveness, but every property that can be expressed in the temporal logics CTL or LTL can be solved in polynomial space.

# 5 Dynamic Restrictions of Eos

In this chapter, we introduce dynamic restrictions for elementary object nets and generalised state machines. Contrary to the structural restrictions introduced in the previous chapter, in which some kind of restrictions to the net's structure was imposed, dynamic restrictions in some manner restrict the possible dynamic behaviour of the net system.

At first, we introduce unary and persistent Eos in Section 5.1. In these systems, conflicts are ruled out dynamically. Both formalisms are shown to be Turing-complete and thus these restrictions are not very helpful for us.

We then introduce safeness for Eos in Section 5.2, i.e. dynamic restrictions with the aim to guarantee finiteness of the state space. We introduce four different variants of safeness where each can be seen as a logical generalisation of the safeness notion for p/t nets. We prove that actually only two of these guarantee finiteness of the state space and that the other two give rise to Turing-complete formalisms.

Even for severe structural restrictions, the reachability problem tends to be hard to decide from an algorithmic point of view. However, for the safe Eos from Section 5.2 that enjoy a finite state space, we show in Section 5.3 that it is not only possible to decide the reachability problem in polynomial space, but that it is possible to do so for every property that can be expressed in the temporal logics CTL or LTL. This generalizes a result by Esparza [Esp98a],

In Section 5.4, we investigate safe Eos that are subject to an additional structural restriction. Contrary to the case of p/t nets for which quite often quicker algorithms for the reachability problem can be devised in such a setting, it turns out that the PSpace-boundary established for safe Eos is in most cases optimal.

The results regarding unary and persistent Eos presented in Section 5.1 have not been published before. However, the proofs require only minor modifications to constructions used before.

Safeness for Eos was introduced by me and Michael Köhler-Bußmeier in [KBH10b], see also [KBH10a].

Also in [KBH10b] is the proof that properties expressed in LTL can be decided in polynomial space for safe Eos.

The theorem establishing a similar statement for properties expressed in CTL was published as joint work with Michael Köhler-Bußmeier in [KBH11b].

The results presented in Section 5.4 combining structural and dynamic restrictions are presented here for the first time.

The tables in Section 5.4 summarise the results obtained in this chapter concerning the complexity of the reachability problem of the various investigated formalisms (see Tables 5.1 and 5.2). However, in the temporal logics LTL and CTL many more problems can be expressed. The results obtained in Section 5.3 are thus much more far-reaching.

## 5.1 Unary and Persistent Eos

In the context of p/t nets, persistent p/t nets and unary p/t nets are known and have been investigated in the past (see the treatment of dynamic restricted Petri nets in Section 2.5).

These notions can be easily transferred to Eos, but the formalisms arising from this restrictions remain Turing-complete.

In the case of a persistent Eos we demand that all transitions that are enabled at a reachable marking indeed can fire. In the case of an unary Eos we demand that at each reachable marking at most one event is enabled. In both cases conflicts are dynamically ruled out. Note that each unary Eos is also persistent and so a Turing-completeness result for unary Eos carries over to persistent Eos.

**Definition 5.1** (**Persistent Eos**). *Let OS be an* Eos *with initial marking* $\mu_0$. *OS is* persistent *or* dynamically conflict-free*, if for each reachable marking* $\mu$ *and every two different events* $\widehat{t}[\vartheta]$, $\widehat{t'}[\vartheta']$, *both activated in* $\mu$, *markings* $\mu'$ *and* $\mu''$ *exist such that* $\mu \xrightarrow{\widehat{t}[\vartheta]} \mu' \xrightarrow{\widehat{t'}[\vartheta']} \mu''$ *holds.*

**Definition 5.2** (**Unary Eos**). *Let OS be an* Eos *with initial marking* $\mu_0$. *OS is* unary*, if in every reachable marking* $\mu$ *at most one event is enabled.*

The above Definition 5.1 ensures that whenever two events are activated in a marking, firing one does not disable the other. This is a dynamical way to state the absence of conflicts and a structural definition should also accomplish this. For p/t nets the definition of persistence captures a broader class of nets than the definition of conflict-freedom, i.e. each conflict-free p/t net is also persistent, while the converse does not hold in general (see Section 2.5). But in exchange conflict-freedom is easily verifiable for p/t nets.

It would be nice to have a similar easily verifiable structural definition of conflict-freedom for Eos. Unfortunately, it turns out that it is unlikely that such a definition exist, due to the labelling of the channels and most importantly nondeterminism in the firing rule itself (see the discussion in Section 4.6).

The definition of persistent and unary Eos restrict the formalism of Eos, but the restrictions are not substantial: the formalisms are still Turing-complete. This can be shown by the same simulation of counter programs as for Eos.

**Theorem 5.3.** *Each counter program can be bisimulated by a unary* Eos.

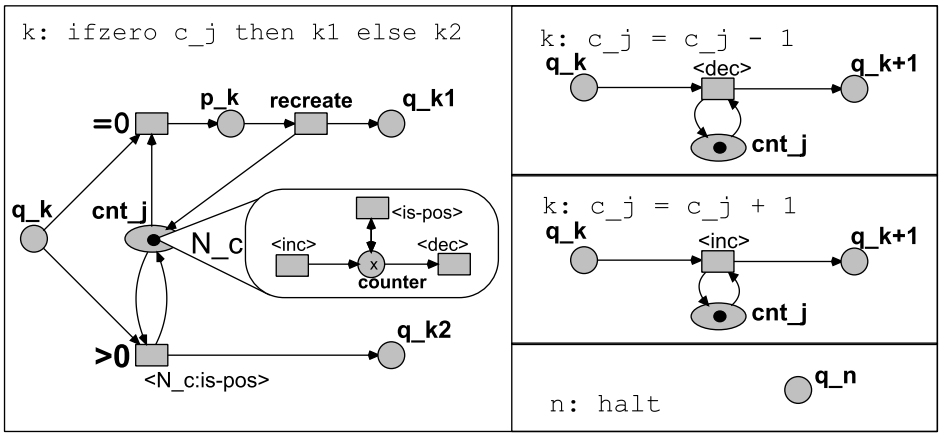


Figure 5.1: The Eos-translation of counter commands (same as Figure 3.5 in Section 3.3)

*Proof.* In the proof of Theorem 3.13 we have shown that the four operations of a counter program can be simulated by an Eos. The Eos-fragments that simulate the different commands are depicted in Figure 5.1 (this is the same figure as Figure 3.5 in Section 3.3 and repeated here for convenience).

The places $q_i$ mark the current position in the counter program, the places $p_k$ are intermediate places between the positions $q_k$ and $q_{k+1}$ in the counter program and are used to remember that the test for zero has succeeded and that a new empty net token has to be created. From all the places $q_i$ and $p_j$ exactly one place is marked with a black token at any reachable marking.

The only possibility where two events could be enabled is when the place $q_k$ corresponding to a jump operation in the counter program. But then the transition =0 is only enabled, if the marking of the object net residing on $cnt_j$ is empty, i.e. when the place *counter* bears no tokens. The transition >0 - or more accurately: the event consisting of the system net transition >0 and the object net transition labeled with the channel *is-pos* - is contrarily only enabled, if the place *counter* is *not* empty.

Thus also in this case only one event is enabled and thus there is at most one event enabled at any reachable marking and the Eos is therefore unary. □

Since every unary Eos is also persistent, persistent Eos are also Turing-complete.

**Corollary 5.4.** *Each counter program can be bisimulated by a persistent* Eos*.*

Thus, even if persistence is a restriction of the general Eos-formalism, this restriction is still powerful enough to simulate Turing machines and thus many important problems (such as reachability) are undecidable. Likewise the restriction to unary Eos is not strong enough to be unable to simulate Turing machines.

## 5.2 Safe Eos and GSM

While the dynamic restrictions introduced in the last section result in Turing-complete formalisms and are thus not truly restricting the formalism of elementary object systems, we turn to one of the arguably most helpful dynamic restrictions for Petri nets now: to the concept of safeness.

Safeness guarantees that the state space of a p/t net is finite and for 1-safe p/t nets, i.e. for nets where on each place at most one token resides, most problems become decidable in polynomial space (see Section 2.5).

In this section, we introduce four different kinds of safeness for Eos as a generalisation of the safeness notion for p/t nets. Two of these safeness notions result in Turing-complete formalisms, while the other two actually guarantee a finite state space.

At first we repeat a result from [Köh07] regarding so called *semi-bounded* Eos.

**Definition 5.5 (Semi-bounded Eos).** *An* Eos *OS is* $N$-bounded *for an object net* $N \in \mathcal{N}$ *if* $|\{\Pi^2_N(\mu) \mid \mu \in R(OS)\}| < \infty$.

*An* Eos *is* semi-bounded *if it is* $N$*-bounded for all object nets* $N \in \mathcal{N}$.

In a semi-bounded Eos all object nets are bounded nets, but it is possible that an Eos is $N$-bounded but $N$ considered as a p/t net in isolation is not bounded for the initial marking $\Pi^2_N(\mu_0)$, i.e. the p/t net system $(N, \Pi^2_N(\mu_0))$ is not bounded. Any unbounded object net where each transition is synchronised with a dead system net transition is an example.

In case of semi-bounded Eos the following theorem holds.

**Theorem 5.6.** *The reachability problem is decidable for semi-bounded* Eos.

Theorem 5.6 is proven in [Köh07]. The main idea in the proof is to construct a simulating p/t net where the set of places consists of tuples $(\widehat{p}, M)$ where $\widehat{p}$ is a system net place and $M$ ranges over all markings of the object net that resides on $\widehat{p}$. This set is finite due to the semi-boundedness of the Eos. The set of transitions coincides with the set of tuples $(\theta, \lambda, \rho)$ where $\theta$ is an event enabled for the mode $(\lambda, \rho)$ in the marking $\mu$ that is encoded in the connections from the places to this transition. Again, since the Eos is semi-bounded there are only finitely many firing modes and thus the set of transitions is finite.

While decidable, the reachability problem for semi-bounded Eos is at least as hard as for p/t nets and thus ExpSpace-hard, since there is no restriction to the system net.

**Theorem 5.7.** *The reachability problem for semi-bounded* Eos *is* ExpSpace*-hard.*

The definition of semi-boundedness from [Köh07] can thus be seen as a generalisation of the safeness notion for p/t nets, but it is not strong enough to

gain better complexity bounds as is the case for p/t nets. This simply stems from the fact that the system net is restricted in no way at all.

We now extend the definition of 1-safe p/t nets to Eos. In a 1-safe p/t net all reachable markings can also be interpreted as sets (of marked places). This is the idea we carry over to Eos.

**Definition 5.8 (Safe Eos).** *Let* $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *be an* Eos.

- *OS is* safe(1) *iff all reachable markings are sets, i.e.*

$$\forall \mu \in R(OS, \mu_0) : \forall \widehat{p}[M] \in M_{\mathcal{N}} : \mu(\widehat{p}[M]) \leq 1$$

- *OS is* safe(2) *iff for all reachable markings there is at most one token on each system net place:*

$$\forall \mu \in R(OS, \mu_0) : \quad \forall \widehat{p} \in \widehat{P} : \Pi^1(\mu)(\widehat{p}) \leq 1$$

- *OS is* safe(3) *iff for all reachable markings there is at most one token on each system net place and each net-token is* 1*-safe:*

$$\begin{aligned} &\forall \mu \in R(OS, \mu_0) : \\ &\quad \forall \widehat{p} \in \widehat{P} : \Pi^1(\mu)(\widehat{p}) \leq 1 \wedge \\ &\quad \forall N \in \mathcal{N} : \forall p \in P_N : \forall \widehat{p}[M] \in M_{\mathcal{N}} : \mu(\widehat{p}[M]) > 0 \Rightarrow M(p) \leq 1 \end{aligned}$$

- *OS is* safe(4) *iff for all reachable markings there is at most one token on each place with respect to projections:*

$$\begin{aligned} \forall \mu \in R(OS, \mu_0) : \quad & \forall \widehat{p} \in \widehat{P} : \Pi^1(\mu)(\widehat{p}) \leq 1 \wedge \\ & \forall N \in \mathcal{N} : \forall p \in P_N : \Pi^2_N(\mu)(p) \leq 1 \end{aligned}$$

*An OS is also called* system-safe*, if it is safe(2),* safe *if it is safe(3), and* strongly safe *if it is safe(4).*

An Eos that is strongly safe, i.e. that is safe(4), is also safe, i.e. safe(3) which is a part of the following lemma.

**Lemma 5.9.** *If an* Eos *is safe(4), then it is safe(3).*
*If an* Eos *is safe(3), then it is safe(2).*
*If an* Eos *is safe(2), then it is safe(1).*

*Proof.* Let $OS$ be a safe(4) Eos with initial marking $\mu_0$ and let $\mu \in R(OS, \mu_0)$. Let $\widehat{p}[M] \in M_{\mathcal{N}}$ with $\mu(\widehat{p}[M]) > 0$. Let $N := d(\widehat{p})$, then $M$ is a marking of an object net of type $N$. Since $OS$ is safe(4) we have $\Pi^2_N(\mu)(p) \leq 1$, which means that even if the markings of all object nets of type $N$ are summed up the place $p$ is marked at most once. This implies $M(p) \leq 1$ and $OS$ is thus also safe(3).

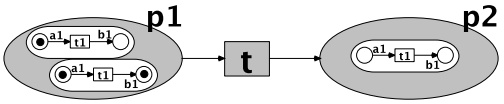


Figure 5.2: An Eos which is safe(1), but not safe(2)

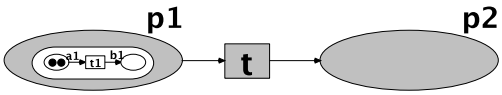


Figure 5.3: An Eos which is safe(2), but not safe(3)

Let $OS$ be a safe(3) Eos, then it is also safe(2) by definition.

Let $OS$ be a safe(2) Eos. Let $\mu$ be a reachable marking and $\widehat{p}[M] \in M_{\mathcal{N}}$. Assume $\mu(\widehat{p}[M]) \geq 2$. Then the place $\widehat{p}$ would be marked with at least two net tokens which implies $\Pi^1(\mu)(\widehat{p}) \geq 2$, a contradiction. Thus $OS$ is also safe(1). □

A safe Eos might thus be safe(3) or safe(4). The reason for this notion is that safe Eos enjoy a finite state space which is proven below (see Theorem 5.11). The term system-safe stems from the fact that the system net places are marked similarly as in a 1-safe net.

Also note that the motivation for the definition of safe(1) Eos was to enforce all reachable markings to be sets as is the case for safe p/t nets. With Lemma 5.9 this is also the case for safe(2), safe(3), and safe(4) Eos.

The converse implications of Lemma 5.9 are in general not true. Examples are given in Figure 5.2, 5.3, and 5.4.

However, the next theorem shows that the notions of safeness introduced in Definition 5.8 coincide for Eos that are essentially p/t nets.

**Theorem 5.10.** *Let* $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ *be an* Eos *with* $d(\widehat{P}) = \{N_\bullet\}$, *i.e. with places for black tokens only. Then safe(1), safe(2), safe(3), and safe(4) are equivalent.*

*Proof.* From Lemma 5.9 we know that a safe(4) Eos is also safe(3), that a safe(3) Eos is also safe(2), and that a safe(2) Eos is also safe(1). It thus suffices to show that a safe(1) Eos with $d(\widehat{P}) = \{N_\bullet\}$ is also safe(4).

Note that $N_\bullet$ has no places (or transitions) and thus $M_{\mathcal{N}} = M_{\{N_\bullet\}}$ can be identified with the set of system net places $\widehat{P}$.

But then $\mu(\widehat{p}[M]) \leq 1$ in the definition of safe(1) means nothing less than that the place $\widehat{p}$ is marked with at most one (black) token in each reachable marking which implies $\Pi^1(\mu)(\widehat{p}) \leq 1$.

Since $P_{N_\bullet} = \emptyset$ the second condition in the definition of safe(4) holds immediately and a safe(1) Eos with $d(\widehat{P}) = \{N_\bullet\}$ is thus also safe(4) which concludes the proof. □

Theorem 5.10 also means that for p/t nets the four different notions of safeness coincides.

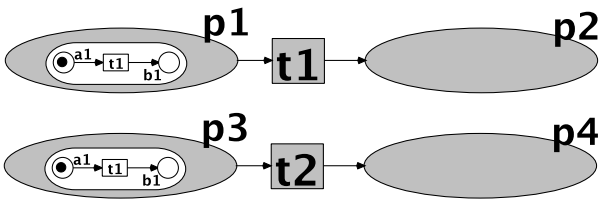


Figure 5.4: An Eos which is safe(3), but not safe(4)

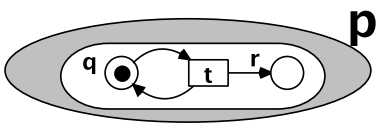


Figure 5.5: Safe(2) Eos with infinite reachability set

Lemma 5.9 and Theorem 5.10 thus give justification for the different notions of safeness given in Definition 5.8: The safeness notions all coincide for p/t nets, but are different for Eos, yet they all capture the idea of "markings being sets" that was capture by safeness for p/t nets.

Moreover, in Theorems 5.11 and 5.12 below it is shown that only for safe(3) and safe(4) Eos the state space is finite, justifying the notion of safeness and strongly safeness.

Note that by definition in the reachable markings of safe(2), safe(3), and safe(4) Eos each system net place is marked with at most one net token. The set $\{\Pi^1(\mu) \mid \mu \in R(OS)\}$ is thus a finite set and appears to be similar to the reachable sets of safe p/t nets. However, an Eos can be safe(4) (and thus safe(3), safe(2), and safe(1)) and yet the system net treated as a p/t net with initial marking $\Pi^1(\mu_0)$ is not safe. Figure 5.6 gives an example, where $\Pi^1(\mu_0) = \widehat{p}_1$ and the set of reachable markings is then $\{\widehat{p}_1 + k\widehat{p}_2 \mid k \in \mathbb{N}\}$.

Nonetheless for safe(2), safe(3), and safe(4) Eos the set $\{\Pi^1(\mu) \mid \mu \in R(OS)\}$ is finite by definition. Furthermore, for safe(3) and safe(4) Eos the net tokens are also safe by definition and thus the set of reachable markings associate with them is finite, too. Although in this case, too, Eos exist which are safe(4), but whose object nets in isolation are not safe (Figure 5.7 shows an example), this combination of finite set of reachable markings associate with the system net and finite set of reachable markings associate with the object nets in the case of safe(3) or safe(4) Eos indeed results in a finite state space for the Eos.

**Theorem 5.11.** *If an* Eos *is safe(3) or safe(4), then its set of reachable markings is finite.*

*Proof.* By Lemma 5.9 an safe(4) Eos is also safe(3), it thus suffices to prove the statement for safe(3) Eos.

Let $OS$ be a safe(3) Eos. Let $k := |\widehat{P}|$ and $l := \max\{|P_N| \mid N \in \mathcal{N}\}$ be the number of system net places and the maximum number of places present in an object net.

Be definition of safe(3) each net token is 1-safe and thus there are at most $2^l$ different markings a net token may have. Also by definition of safe(3) each

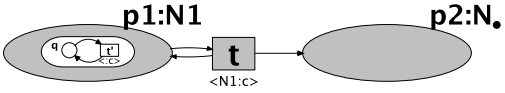


Figure 5.6: An Eos which is safe(4), but where $(\widehat{N}, \Pi^1(\mu_0))$ is an unsafe p/t net

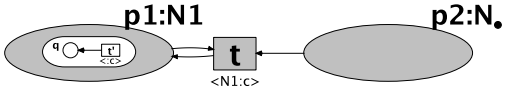


Figure 5.7: An Eos which is safe(4), but with an unsafe object net, if treated in isolation

system net place is either marked or unmarked with a net token with one of these markings, thus there are up to $2^l + 1$ different possibilities a system net place may be marked with. Since there are $k$ system net places, this results in at most $(1 + 2^l)^k$ different markings of $OS$, i.e. $|R(OS)| \leq (1 + 2^l)^k$. □

The property of a finite state space does no longer exist for safe(1) or safe(2) Eos. The Eos in Figure 5.5 is safe(2) and thus also safe(1) due to Lemma 5.9, but its set of reachable markings is $R(OS) = \{\widehat{p}[q + k \cdot r] \mid k \in \mathbb{N}\}$ and thus infinite.

Not only do safe(1) and safe(2) Eos have, in general, state spaces of infinite size, due to the unbounded net tokens, but the net tokens can also be used to encode counters and thus to prove undecidability results.

**Theorem 5.12.** *Each inhibitor net can be simulated by a safe(1) or safe(2)* Eos *that is in addition strongly deterministic.*

*Proof.* The same construction as in the proof of Theorem 3.14 can be used. On each system net place resides at most one net token and thus the constructed Eos is safe(2) and also safe(1). Since moreover the same adjustments as in Theorem 4.12 are possible, the constructed Eos is even strongly deterministic. □

**Corollary 5.13.** *Reachability, liveness, coverability, and boundedness are undecidable for safe(1) or safe(2)* Eos *even if the* Eos *is strongly deterministic.*

A similar result holds also for *conservative* and safe(1) or safe(2) Eos:

**Theorem 5.14.** *Reachability and liveness are undecidable for conservative and safe(1) or safe(2)* Eos *even if the* Eos *is strongly deterministic.*

*Proof.* In the proofs of Lemma 4.3 and Lemma 4.4 the constructed Eos are not only conservative but also safe(2), since there is at most one net token on each system net place. Due to Lemma 5.9 these Eos are then also safe(1) and since the same modification as in Theorem 4.12 is possible (see also Theorem 4.13), the Eos is even strongly deterministic. □

To summarise, we have introduced four different variants of safeness and justified them. By Theorem 5.11 in general only safe(3) and safe(4) Eos have

**Algorithm 1** `IsEOSSafe`($OS$). Check if $OS$ is safe.

1: $\mu \leftarrow \mu_0$
2: **for** $i = 1$ **to** $(1 + 2^l)^k + 1$ **do**
3: choose nondeterministically an $\theta = (\widehat{t}, t_1, \ldots, t_{|\mathcal{N}|}) \in \widehat{T} \times T_1 \times \ldots \times T_{|\mathcal{N}|}$
4: **if** $\theta \in \Theta$ **then**
5: choose nondeterministically a mode $(\lambda, \rho)$ such that $\mu \xrightarrow{\theta(\lambda,\rho)}$
6: calculate $\mu'$ with $\mu \xrightarrow{\theta(\lambda,\rho)} \mu'$
7: **if** $\mu'$ is not safe(3) **then**
8: **return true**
9: **end if**
10: $\mu \leftarrow \mu'$
11: **end if**
12: **end for**
13: **return false**

a finite state space, i.e. the safe and strongly safe Eos, while safe(1) and safe(2), i.e. the system safe Eos are Turing-complete.

Whether an Eos is safe, can be decided in polynomial space.

**Theorem 5.15.** *Given an* Eos *OS it is* PSpace*-complete to decide if OS is safe.*

*Proof.* At first PSpace-hardness follows from the result for 1-safe p/t net systems, see Theorem 2.52.

Due to Theorem 5.11 the state space of a safe Eos is bounded and due to the proof this bound is $(1 + 2^l)^k$ with $k := |\widehat{P}|$ and $l := \max\{|P_N| \mid N \in \mathcal{N}\}$.

If $OS$ is *not* safe, then among all computations of length $k \leq (1 + 2^l)^k + 1$ there must be at least one in which a marking occurs which does not fulfil the conditions of a safe(3) Eos. Otherwise in all these computations a marking appears twice and thus the computation loops - or has already ended in a deadlock. All reachable markings thus satisfy the conditions of a safe(3) Eos and thus $OS$ is safe(3).

Algorithm 1 applies this idea and returns `true` if the Eos given as input is *not* safe.

In the algorithm it is easily possible to check if $\theta$ is an event or not by inspecting the labelling functions. Then, since $\mu$ is a safe(3) marking, in the possible modes $(\lambda, \rho)$ the submarking $\lambda$ is unique. Different markings $\rho$ are possible but only due to the distribution of the marking of a net token type to different net tokens of that type. Of these one can be guessed again and $\mu'$ can be computed. These computations and the test if $\mu'$ is safe(3) are easily possible in space polynomial in the size of $OS$. The number $i$ also requires only space in $O(l \cdot k)$ and thus in space polynomial in the input.

Altogether Algorithm 1 is an NPSpace-algorithm. With the result from Savitch [Sav70] this algorithm can be transformed into a deterministic PSpace-algorithm whose answer can be reversed such that the resulting algorithm returns `true` if and only if *OS* is a safe Eos. □

For generalised state machines there are actually fewer safeness notions with regard to Definition 5.8. Indeed, for a GSM the notions of safe(1) and safe(2) and the notions of safe(3) and safe(4) coincide. Lemma 5.9 can thus be strengthened considerably:

**Theorem 5.16.** *Let $G$ be a GSM, then $G$ is safe(1) iff it is safe(2) and $G$ is safe(3) iff it is safe(4).*

*Proof.* By Lemma 5.9 an safe(4) Eos is also safe(3) and a safe(2) Eos is also safe(1), thus only the converse implications need to be proven.

Assume that the GSM is safe(1). By definition of a GSM at most one net token of type $N \in \mathcal{N} \setminus \{N_\bullet\}$ exists in the initial marking and thus due to the structure of a GSM by induction also in any reachable marking. For places typed with an $N \in \mathcal{N} \setminus \{N_\bullet\}$ the condition for safe(2) is thus fulfilled.

Now let $\widehat{p} \in \widehat{P}$ with $d(\widehat{p}) = N_\bullet$. For $N_\bullet$ the set $\{\widehat{p}\} \times M_{N_\bullet}$ can be identified with $\widehat{p}$ and thus $\mu(\widehat{p}[M]) \leq 1$ implies $\Pi^1(\mu)(\widehat{p}) \leq 1$ from which it follows that $G$ is also safe(2).

Now assume that the GSM is safe(3). Again, since only one net token of type $N \in \mathcal{N} \setminus \{N_\bullet\}$ exists in any reachable marking $\mu(\widehat{p}[M]) > 0$ implies that the net token of type $d(\widehat{p})$ currently residing on $\widehat{p}$ and marked with $M$ is the only net token of this type and thus $M(p) \leq 1$ for $p \in P_{d(\widehat{p})}$also implies $\Pi^2_N(\mu)(p) \leq 1$ and thus $G$ is also safe(4). □

As was possible for Eos it is also possible to decide if a given GSM is safe or not. Furthermore, it can also be tested if given Eos is actually a safe GSM by applying Theorem 5.15 and doing some more structural checks.

**Theorem 5.17.** *Given a GSM $G$ it is* PSpace*-complete to decide, if $G$ is safe. Given an* Eos *OS it is* PSpace*-complete to decide if OS is a safe GSM.*

Note that in a safe(1) or safe(2) GSM the only restriction is on the system net places that are typed with $N_\bullet$, i.e. with the system net places that may be marked with black tokens. These places are restricted in such a way that they may at most be marked with one black token.

In the context of generalised state machines it is thus sensible to either speak of a GSM, a *system-safe GSM*, if it is safe(1) or safe(2), or of a *safe GSM*, if it is safe(3) or safe(4), i.e. there is at most one net token on each system net place each of whom is either a black token or a safe p/t net. Note that the examples in Figure 5.6 and 5.7 are still valid. These are safe GSMs where the system net or an object net treated in isolation are not safe.

Moreover, since in a system-safe GSM an object net can be an arbitrarily p/t net, system-safe GSM are still as problematic as p/t nets in terms of complexity. Thus the notion of system-safeness for GSMs is only sensible in combination with other restrictions.

In the following we will focus on safe(3) and safe(4) Eos and GSM for which due to their finite state space reachability, liveness, and so on are decidable. The problem is that compared to the state spaces of safe p/t nets their state space may become quite large: While for 1-safe p/t nets with $n$ places the number of reachable markings is bound by $2^n$, in an safe(3) or safe(4) Eos where the number of places in the system net and in all object nets is also bound by $n$, the number of reachable states is bounded by $(1 + 2^n)^n$ (see Theorem 5.11). Thus if the system net and the object nets have $n = 10$ places and are thus quite small, they have a state spaces of size $2^{10}$ if considered in isolation, which still is small enough to be directly represented and analysed. Composed in an Eos the state space's size might go beyond $2^{100}$ states, however, which makes it in general hard if not impossible to represent the state space explicitly.

Nonetheless despite the huge state space it is possible to apply model checking techniques to safe(3) and safe(4) and otherwise unrestricted Eos and to decide every property expressible in the temporal logics CTL or LTL in space only polynomial in the size of the Eos. This surprising result is the topic of the next section.

## 5.3 LTL and CTL Model Checking of Safe Eos

From the four different notions of safeness introduced in the last section (see Definition 5.8) we concentrate on safe(3) Eos, simply called safe Eos, in the following. Since safe(1) and safe(2) Eos are Turing-complete due to Theorem 5.12 the reachability problem can not decided for them. On the other hand any positive result for safe(3) Eos carries over to safe(4) Eos, because a safe(4) Eos is also a safe(3) Eos due to Lemma 5.9.

Instead of focussing on the reachability problem as in Chapter 4 where structural restrictions are treated, we prove in this section that not only reachability but indeed every property that can be stated in the temporal logics CTL or LTL can be decided for a safe Eos and thus also for a safe GSM.

Since safe Eos have a finite state space due to Theorem 5.11, decidability itself is not too surprising, since most problems can be decided by a brute force algorithm. But despite the possibly huge number of events of an Eos apparent in Lemma 3.10 and Lemma 4.10 and the enormously sized state space even of safe Eos apparent in the proof of Theorem 5.11, properties expressed in CTL or LTL can not only be decided, but for the decision procedure space polynomial in the size of the input, i.e. in the size of the Eos, suffices. The procedure presented here is a generalisation of the algorithms and methods from [Esp98a],

which focuses on 1-safe p/t nets. While the methods from [Esp98a] can be easily translated to the setting of Eos here, it is necessary to prove for each step, that the step is still possible despite the more complicated firing rule, the possible enormous state space, and so on.

At first it is a known fact that most interesting questions about the behaviour of classical 1-safe p/t nets like liveness, deadlock-freedom, and reachability are PSpace-hard (see [Esp98a, JLL77]). Since 1-safe p/t nets can be seen as a special kind of safe(4) Eos, which are also safe(3), safe(2), and safe(1), these results directly carry over to safe Eos.

**Theorem 5.18.** *Reachability, liveness, and coverability are* PSpace*-hard for safe(4) or safe(3) and strongly deterministic* Eos*.*

*Proof.* All problems mentioned are PSpace-hard for 1-safe p/t nets due to Theorem 2.52. Given a 1-safe p/t net system $(N, m_0)$, an Eos $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ that bisimulates it can be easily constructed. The system net corresponds to the given p/t net, i.e. $\widehat{N} := N$. The only object net type is the object net $N_\bullet$ used to model black tokens. All places are labeled with this object net type and all transitions are labelled with the label for system autonomous firing, i.e. there are no synchronous events. The initial marking $\mu_0$ corresponds to $m_0$: If $m_0(p) = 1$, then $\mu_0(p[\mathbf{0}]) = 1$.

Clearly if a transition $t$ is enabled in $N$, the system autonomous event $t[]$ is also enabled in $OS$ and firing $t[]$ results in a marking that correspond to the marking reached after firing $t$ in $N$.

Since $N$ is 1-safe, on each system net place of $OS$ resides at most one net token, which are all of the same type, namely $N_\bullet$, and which are all marked with $\mathbf{0}$, so the sum of these markings is $\mathbf{0}$, too, and $OS$ is safe(4).

Due to this construction, which is possible in polynomial time, the hardness results for 1-safe /t nets from Theorem 2.52 carry over to safe(4) Eos and thus to safe(3) Eos, too. □

The theorem above can be generalised to transfer further hardness results from 1-safe p/t nets to safe(4) or safe(3) and strongly deterministic Eos.

However, the more interesting question is, if polynomial space suffices. In the following two sections, we will show that CTL and LTL model checking is possible in polynomial space for safe(3) Eos, i.e. given a safe(3) Eos $OS$ we establish a notion of when $OS$ satisfies a formula $\phi$, denoted by $OS \models \phi$ and then show that given $OS$ and $\phi$ it is possible to decide in polynomial space if $OS \models \phi$ holds.

For this analysis it is necessary to have a look at fundamental properties of Eos again and characterise them in terms of complexity. For example from Theorem 3.10 and the discussion preceding and following it is evident that the number of events of an Eos $OS$ is exponential in the size of $OS$. It is thus not

possible to keep them all in memory if certain low complexity bounds should be met.

This will be done in various lemmata throughout the subsections about LTL and CTL model checking safe Eos below.

## LTL Model Checking of Safe Eos

In the following we show that it is possible to decide given a safe Eos and an arbitrary LTL formula, specifying a property of the Eos, if the Eos satisfies the property or not, using only polynomial space in the size of the Eos and the size of the formula.

Since an instance of the reachability problem can be expressed with an LTL formula, it follows that the reachability problem for safe Eos is PSpace-complete.

The technique used is a generalisation of the method presented in [Esp98a] for 1-safe p/t nets which in turn heavily relies on the results from [Var96]. We now briefly sketch the method from [Esp98a] to verify if a 1-safe p/t net satisfies a property expressed in LTL.

Let $\phi$ be an LTL formula and $N$ a 1-safe p/t net. $N$ satisfies $\phi$ if the LTS that is obtained from the reachability graph of $N$, satisfies $\phi$, where the nodes of the reachability graph are labelled with the places that are marked in that nodes' marking. The set of propositions is chosen accordingly to consists of the places of $N$.

To prove if $N$ satisfies $\phi$ two automata, a finite automaton $A_\phi$ and a Büchi automaton $B_\phi$, are used, where $L(A_\phi) \cup L_\omega(B_\phi)$ is exactly the set of computations satisfying the formula $\phi$. In the case of 1-safe p/t nets a computation is a sequence of sets of places and the atomic propositions of $\phi$ are therefore the places of the net $N$ and the alphabet of both $A_\phi$ and $B_\phi$ is $2^P$. The construction of these automata exceeds the scope of this work (see [Esp98a] and [Var96] for details). Here it suffices to know the following two facts: First, the states of $A_\phi$ are sets of subformulas of $\phi$ and the states of $B_\phi$ are pairs of sets of subformulas of $\phi$, thus they both may have exponentially many states in $|\phi|$ and must not be completely constructed if a polynomial space bound should be met. Yet a single state has polynomial size in $|\phi|$. Second, given two states $q_1$ and $q_2$ of $A_\phi$ or $B_\phi$ and a marking $\mu$ (which is a symbol of the alphabet of $A_\phi$ resp. $B_\phi$), it is possible to decide in polynomial space (in $|\phi|$) if a transition from $q_1$ to $q_2$ with label $\mu$ exists.

Two further automata, $A_N$ and $B_N$, are used, obtained from the safe Eos. The set of states of $A_N$ and $B_N$ correspond to the reachable markings of the net $N$. Indeed, $A_N$ and $B_N$ correspond closely to the reachability graph and only differ from it in the edges' labels. The initial state of $A_N$ and $B_N$ is the initial marking and the transition relation $\delta_N$ (again the same for $A_N$ and $B_N$) contains the triple $(m, m, m')$ of markings if $m'$ can be reached from $m$ via some

transition $t$. The set of final states of $A_N$ is the set of deadlocked reachable markings of $N$ and the set of final states of $B_N$ contains all reachable markings of $N$, thus $L(A_N)$ is the set of all finite and $L_\omega(B_N)$ is the set of all infinite runs of $N$. Again $A_N$ and $B_N$ may have exponentially many states in $|N|$ and again can not be constructed completely.

Then two automata $A$ and $B$ are defined with the usual construction to be the product automaton of $A_{\neg\phi}$ and $A_N$, resp. of $B_{\neg\phi}$ and $B_N$. Then $L(A) = L(A_{\neg\phi}) \cap L(A_N)$ and $L_\omega(B) = L_\omega(B_{\neg\phi}) \cap L_\omega(B_N)$ holds. While in general it is not the case that the product of two Büchi automata accepts the intersection of the languages, it indeed holds here. Now $L(A) \cup L_\omega(B)$ is the set of runs of $N$ that do *not* satisfy $\phi$. The question whether $N$ satisfies $\phi$, i.e. if all of $N$'s runs satisfy $\phi$, is thus reduced to the question if $L(A)$ and $L_\omega(B)$ are both empty.

As mentioned, the construction of $A$ and $B$ is the usual product construction and thus does not cause problems. Unfortunately, $A_\phi$ and $B_\phi$ may have exponentially many states in $|\phi|$ and $A_N$ and $B_N$ may have exponentially many states in $|N|$. But $A$ and $B$ can be constructed on the fly and only a constant number of states needs to be saved. The algorithm mainly guesses, given a state $(q, M)$, a possible next state $(q', M')$ in $A$, resp. $B$, checks if the transition from $(q, M)$ to $(q', M')$ is indeed possible and, if so, enters this state. The algorithm for $A$ accepts if it enters a final state of $A$. For $B$ the case is more complicated, since $B$ only accepts if it enters a final state infinitely often. To achieve this, the algorithm for $B$ guesses that the current final state is revisited and checks this. The case for $B$ is thus only slightly more involved. These algorithms solve the nonemptiness problem, which by reversing the answer solves the above mentioned emptiness problem.

Note that the algorithms described above are nondeterministic, but run in polynomial space, since the only four things important are

1. Given two states $q_1, q_2$ of $A_\phi$ or $B_\phi$ and a marking $M$, it is possible to decide in polynomial space (in $|\phi|$) if a transition from $q_1$ to $q_2$ with label $M$ exists.

2. Given two markings $M$ and $M'$ and a transition $t$, it is possible to decide in polynomial space (in $|N|$) if $M \xrightarrow{t} M'$ holds.

3. It is possible to decide in polynomial space (in $|N|$ and $|\phi|$) if a state of $A$ or $B$ is a final state.

4. A state of $A$ or $B$ has polynomial size (in $|N|$ and $|\phi|$).

The last three items hold immediately for 1-safe p/t nets (the size of a state of $A_\phi$ or $B_\phi$ is at most quadratic in $|\phi|$). The first item follows from [Var96] on which the above described method from [Esp98a] heavily relies.

Now, since this algorithm runs in nondeterministic polynomial space, an deterministic polynomial space algorithm follows by Savitch's construction [Sav70].

This approach can be adapted to the case of safe Eos. In the following explanation the automata constructed are denoted by $A_{OS}$ and $B_{OS}$ instead of $A_N$ and $B_N$ and a marking is denoted by $\mu$ instead of $M$.

At first the set of atomic propositions needs to be adjusted. Let

$$prop := \{\widehat{p}[x] \mid \widehat{p} \in \widehat{P}, x \in \{\mathbf{0}, *\}\} \cup \{\widehat{p}[p] \mid \widehat{p} \in \widehat{P}, p \in P(d(\widehat{p}))\} \cup \{\mathbf{0}\}.$$

The proposition $\widehat{p_i}[\mathbf{0}]$ is used to code that an empty object net resides on the system net place $\widehat{p_i}$. $\widehat{p_i}[*]$ is used to code that the place $\widehat{p_i}$ is marked in *some* way, but it is not important how exactly. The notion $\widehat{p_i}[p_j]$ is used to describe that a net token resides on place $\widehat{p_i}$ and that the place $p_j$ of this net token is marked. Subsequently, $\widehat{p_i}[p_{j_1}] \wedge \widehat{p_i}[p_{j_2}]$ is used to encode that on the system net place $\widehat{p_i}$ an object net resides and that the places $p_{j_1}$ and $p_{j_2}$ of the object net are marked. $\widehat{p_i}[p_{j_1}] \wedge \widehat{p_i}[p_{j_2}]$ thus encodes the (sub-)marking $\widehat{p_i}[p_{j_1} + p_{j_2}]$. Since in an safe Eos on each system net place resides at most one net token, no ambiguity can arise with the marking $\widehat{p_i}[p_{j_1}] + \widehat{p_i}[p_{j_2}]$ which would encode that *two* object nets reside one place $\widehat{p_i}$ with different inner markings.

The syntax of LTL (see Definition 2.19) is restricted to the formulas obtainable by restricting the set of connectives to the adequate set $\{\neg, \wedge, X, U\}$ (see the discussion of adequate sets in Section 2.3).

The semantic is defined similarly as described in Definition 2.17. Differences are in the labelling function of the labelled transition system and the treatment of the atomic propositions.

The labelled transition system which is used to state if an Eos $OS$ satisfies a LTL formula, is the reachability graph $RG(OS)$ of $OS$, that is the set of states is the set of reachable markings, the start state is the state that corresponds to the initial marking $\mu_0$, and the transition relation $R$ is obtained from the edges of $RG(OS)$ by ignoring their labelling. In this setting $R$ is not necessarily left-total, but this is no severe restriction, since one could either add an "error state" to which all states which have no outgoing arcs are linked or one could deal with these states in the algorithms explicitly. In the following we will not distinguish between a marking $\mu$ of an Eos $OS$ and the node in the reachability graph of $OS$ that corresponds to $\mu$.

Different from the description of LTL in Section 2.3 the nodes are not labelled with a subset of the set of atomic propositions *prop* but simply with the marking $\mu$ of that node. If an atomic formula $p \in prop$ holds in $\mu$ is then explicitly computed with the following semantic:

**Definition 5.19** (**Semantic of LTL for safe Eos**). *Let $OS$ be a safe* Eos*, let $p \in prop$ be an atomic proposition as introduced above, let $w$ be a path in*

*the reachability graph of $OS$, and let $\mu \in R(OS)$ be a node in that graph and the first node of $w$. To define if $OS, w \models p$ holds four cases are distinguished:*

1. *$p = \mathbf{0}$. Then $OS, w \models p$ holds if $\mu$ is the empty marking, i.e. if $\mu = \mathbf{0}$.*
2. *$p = \widehat{p_i}[\mathbf{0}]$, $\widehat{p_i} \in \widehat{P}$. Then $OS, w \models p$ holds if an empty net token resides on the place $\widehat{p_i}$ in $\mu$*
3. *$p = \widehat{p_i}[*]$, $\widehat{p_i} \in \widehat{P}$. Then $OS, w \models p$ holds if a net token resides on the place $\widehat{p_i}$ in $\mu$ that is marked arbitrarily.*
4. *$p = \widehat{p_i}[p_j]$, $\widehat{p_i} \in \widehat{P}$ and $p_j \in P_{d(\widehat{p_i})}$. Then $OS, w \models p$ holds if a net token $N$ resides on the place $\widehat{p_i}$ in $\mu$ and in the current marking $M$ of $N$ the place $p_j$ is marked.*

*The other cases, i.e. the formulas $\neg\phi$, $\phi_1 \wedge \phi_2$, $X\phi$, and $E[\phi_1 U \phi_2]$ are treated exactly as in Definition 2.17.*

*If $\phi$ is a LTL formula, we denote by $OS, \mu \models \phi$ that $OS, w \models \phi$ holds according to the inductive definition of $\models$ defined above for every path $w$ that starts at $\mu$ in $RG(OS)$*

*$\phi$ is* satisfied *by $OS$ if $OS, \mu_0 \models \phi$ holds. In this case, $OS$ is a* model *for $\phi$ which is also shortly denoted by $OS \models \phi$.*

A computation is now a sequence of subsets of *prop* again. To exactly describe a marking, it might be necessary, however, to use each variable from *prop* which means that a marking is now quadratic in the size of the net, instead of linear, but it turns out that this is not problematic.

Given *prop* the construction of $A_\phi$ and $B_\phi$ is possible as before, since the change in the size of *prop* from linear to quadratic in the size of the Eos does not matter here. In particular because it is not necessary to fully store the alphabet of $A_\phi$ or $B_\phi$. The states of $A_\phi$ and $B_\phi$ are still quadratic in the size of $|\phi|$ and given two states $q_1, q_2$ of $A_\phi$ or $B_\phi$ and a marking $\mu$ of Eos, it is still possible to decide in polynomial space (in $|\phi|$) if a transition from $q_1$ to $q_2$ with label $\mu$ exists. The size of $\mu$ and thus the size of a state of $A_{OS}$ and $B_{OS}$ is now quadratic in $|OS|$, but this is not a problem, since on the one hand $\mu$ actually remains a constant during this test, and on the other hand is still in polynomial space. Thus, the first and fourth item above are also satisfied by a safe Eos.

The two interesting points thus are, if it is possible to check in the required space bounds whether a final state in $A$ (resp. $B$) is reached and if it is possible to check $\mu \xrightarrow[OS]{\widehat{t}[\theta]} \mu'$ given two markings $\mu, \mu' \in \mathcal{M}$ of $OS$ and an event $\widehat{t}[\theta] \in \Theta$.

Since $A$ and $B$ have been defined via a product construction their final states are tuples consisting of one final state of $A_\phi$ and one final state of $A_{OS}$, resp. one final state of $B_\phi$ and one final state of $B_{OS}$. To check for final states in $A_\phi$ and

$B_\phi$ is very simple, since by their definition one only needs to do some syntactic checks on the states, which are actually sets of subformulas of $\phi$. To check for final states in $B_{OS}$ is also simple, since the set of final states of $B_{OS}$ is exactly the set of reachable markings of $OS$. Since all states we encounter are reachable, this check is trivial. It remains to check if a given state is a final state in $A_{OS}$. For this it is necessary to check if a marking $\mu$ of $OS$, which is a state of $A_{OS}$, is a deadlock, i.e. no event is enabled in $\mu$.

**Lemma 5.20.** *To check if a marking $\mu$ of a safe* Eos *$OS$ is a deadlock, is possible in space polynomial in the size of $OS$.*

*Proof.* Given $\mu$, each transition is checked for autonomous firing. This can be done in a similar manner as for p/t net systems. Different from there, it might be necessary to test an object net transition $t$ of an object net $N$ multiple times, since on each system net place of type $N$ a net token might reside whose transition $t$ is activated. Since, one only has to iterate through all system net places, this is still possible in polynomial time and space.

Then the system net transitions are again investigated for synchronous events. If a system net transition $\widehat{t}$ is fixed the possible synchronous events can be generated one after another using the labelling and the object nets in the preset of $\widehat{t}$. If then $\mu$ and the event are fixed, the possible modes $(\lambda, \rho)$ only differ in the choice of $\rho$ due to the possible distribution of the markings of net tokens of the same type. These can be easily enumerated, but are actually not important here, since one event that is enabled in any mode is enough.

The whole calculation can be done in polynomial space by reusing the space for each event constructed. □

It remains to show that $\mu \xrightarrow[OS]{\widehat{t}[\theta]} \mu'$ can be tested, given two markings $\mu, \mu' \in \mathcal{M}$ and an event $\widehat{t}[\theta] \in \Theta$.

**Lemma 5.21.** *Given a safe* Eos *$OS$, two markings $\mu$ and $\mu'$, and an event $\widehat{t}[\theta] \in \Theta$, it is possible to decide in polynomial space in $|OS|$ if $\mu \xrightarrow[OS]{\widehat{t}[\theta]} \mu'$ holds.*

*Proof.* Since $OS$ is a safe Eos and thus on each system net place resides at most one net token, the submarking $\lambda$ can be calculated from the given marking $\mu$ and the event $\widehat{t}[\theta]$ according to the enabling predicate (see Definition 3.7). Indeed, in $\lambda$ exactly the net tokens in the preset of $\widehat{t}$ are present and $\Pi^2_N(\lambda) \geq \mathbf{pre}_N(\vartheta(N))$ has to hold for every $N \in \mathcal{N}$. The possible modes $(\lambda, \rho)$ then only differ in $\rho$ and only in the distribution of the tokens there to net tokens of the same type but on different places in the postset of $\widehat{t}$. Using the supposed successor marking $\mu'$ one can calculate the correct mode, if any exist. Since the whole procedure only needs some extra space for temporal variables, it can be implemented in the stated space bound. □

**Algorithm 2** Checking $A$ for nonemptiness

**Ensure:** $q$ of type state of $A_{\neg\phi}$
**Ensure:** $\mu$ of type state of $A_N$
1: $(q, \mu) \leftarrow (q_0, \mu_0)$;
2: **while** $(q, \mu)$ is not a final state of $A$ **do**
3: guess a state $q'$ of $A_{\neg\phi}$ with $q \xrightarrow[A_{\neg\phi}]{\mu} q'$
4: guess a marking $\mu'$ and an event $\widehat{\tau}[\theta]$ with $\mu \xrightarrow[OS]{\widehat{\tau}[\theta]} \mu'$
5: $(q, \mu) \leftarrow (q', \mu')$
6: **end while**
7: **return true**

As was the case for 1-safe p/t net systems two automata $A$ and $B$ are defined as the product automata of $A_{\neg\phi}$ and $A_{OS}$ and of $B_{\neg\phi}$ and $B_{OS}$, respectively.

The procedure to check $A$ for nonemptiness is outlined in Algorithm 2 as an adaption of the algorithm given in [Esp98a] to the case of safe Eos.

Apart from using nondeterminism another trick is used: Since $A$ and $B$ may have exponentially many states in $|OS|$ and $|\phi|$, they are constructed *on the fly*. Starting with the initial state as the current state $(q, \mu)$, a next state $(q', \mu')$ is repeatedly guessed. Then it is checked if $(q', \mu')$ is indeed reachable from $(q, \mu)$ via some event and, if so, the current state is updated. In the case of $A$, $L(A)$ is nonempty, if a final state is reached. In this case *true* is returned.

The case of the automaton $B$ is only slightly more complicated. Since $L_\omega(B)$ is nonemtpy if and only if a final state is visited infinitely often, i.e. if there is a reachable final state $q$ such that there is a loop from $q$ to itself, one proceeds as in Algorithm 2, but if a final state is reached, a guess is made that this is the final state that will be revisited or not. This guess is then checked in a similar way as Algorithm 2 checks for any final state. See [Esp98a] for details.

The results and explanations above lead to the following theorem:

**Theorem 5.22.** *Given a safe(3) or safe(4)* Eos *OS and an LTL formula $\phi$, checking whether OS satisfies $\phi$ can be done in polynomial space in the size of OS and $\phi$, that is, there is a polynomial p, independent of OS and $\phi$, such that the algorithm uses $O(p(|OS| + |\phi|))$ space.*

*Proof.* The statement follows from Lemmata 5.20 and 5.21 and the observation that only a constant number of states need to be saved and that the states of $A_{\neg\phi}$ (resp. $B_{\neg\phi}$) and $A_N$ ($B_N$) have polynomial size in $|\phi|$ and $|N|$.

Since $A$ and $B$ have been constructed by use of the product construction, one actually needs to check if a transition is possible in $A_{\neg\phi}$ and $A_{OS}$ to check if one is possible in $A$ (and similar for $B$). For $A_{\neg\phi}$ (and $B_{\neg\phi}$) this follows from the construction of $A_{\neg\phi}$ which is not shown here. (See for example [Var96].) In the algorithm one only needs to store a state of $A_{\neg\phi}$ and a symbol of the

alphabet (which is a marking of $OS$) of $A_{\neg\phi}$. Then one can check similar to Algorithm 2 if a final state can be reached. This can be done in polynomial space. See [Var96] for more details.

To check if a transition in $A_{OS}$ or $B_{OS}$ is possible, a test is necessary if, given two markings $\mu, \mu' \in \mathcal{M}$ and an event $\widehat{\tau}[\theta] \in \Theta$, the relation $\mu \xrightarrow[OS]{\widehat{\tau}[\theta]} \mu'$ holds. This is proven in Lemma 5.21. In Algorithm 2 an event is guessed. An event can be seen as an element of $\widehat{T} \times T_1 \times \ldots \times T_{|\mathcal{N}|}$, which can be easily guessed, but it is then necessary to check if this is indeed an event. This, however, can be easily done with a couple of table lookups using the labelling function given as input.

Finally a test is needed if a state in $A$ (resp. $B$) is a final state, that is a test for final states in $A_{\neg\phi}$, $B_{\neg\phi}$, $A_{OS}$ and $B_{OS}$. For $A_{\neg\phi}$ and $B_{\neg\phi}$ by their definition it is only necessary to do some syntactic checks on the states, which are actually sets of subformulas of $\phi$. This is easily possible in polynomial space. To check for final states in $B_{OS}$ is also simple, since the set of final states of $B_{OS}$ equals its set of states. At last, to check if a state of $A_{OS}$ is a final state, we need to check if a marking $\mu$ of $OS$ is a deadlock. This is proven in Lemma 5.20. □

Since reachability can be expressed as an LTL formula we have the following corollary:

**Corollary 5.23.** *The reachability problem for safe* Eos *is* PSpace*-complete.*

Even if reachability as one of the major problems can thus be decided in polynomial space, other important problems can not be solved with the result from Theorem 5.22. Most prominently the liveness problem for safe Eos can not be solved in this way, because liveness can not be expressed in LTL. It can, however, be expressed with a CTL formula and in the next section it is shown how not only every property expressed in LTL but also every property expressed in CTL can be decided for a safe Eos using only polynomial space in the size of the formula and the Eos.

## CTL Model Checking of Safe Eos

While reachability can be expressed as a LTL formula and it is thus PSpace-complete to decide the reachability problem for safe Eos due to Theorem 5.22, other problems that are not expressible in LTL remain unsolved.

Similar to the case of LTL formulas treated in the previous section, we show in the following that polynomial space is also sufficient to decide if a safe Eos satisfies a property specified in the temporal logic CTL. Since liveness can be expressed in CTL, it follows that the liveness problem for safe Eos is PSpace-complete, too.

Our presentation is again based on ideas from [Esp98a], which focuses on 1-safe p/t nets and uses a different set of CTL connectives.

In the treatment of CTL in Section 2.3 it is stated that CTL formulas are interpreted with respect to states of a transition system or alternatively on possibly infinite computation trees, where each node represents a state of the computation and is labeled with the set of atomic propositions that hold in that state. In the setting here the linear transition system which is used as a model is the reachability graph of the Eos and the tree is simply the unfolding of this reachability graph into a tree.

For the atomic propositions the set used for LTL model checking can be reused. Here we also take each event as an atomic proposition. In this way, the formula to describe if an event can be enabled is shorter. Altogether we define

$$prop := \{\widehat{p}[x] \mid \widehat{p} \in \widehat{P}, x \in \{\mathbf{0}, *\}\} \cup \{\widehat{p}[p] \mid \widehat{p} \in \widehat{P}, p \in P(d(\widehat{p}))\} \cup \{\mathbf{0}\} \cup \Theta.$$

The syntax of CTL (see Definition 2.19) is restricted to the formulas obtainable by the following grammar:

$$\begin{aligned} \phi \quad ::= \quad & p \mid \neg\phi \mid (\phi \wedge \phi) \mid \\ & EX\phi \mid EG\phi \mid E[\phi U \phi] \end{aligned}$$

where $p \in prop$. That is the set of connectives is restricted to the adequate set $\{\neg, \wedge, EX, EG, EU\}$ (see the discussion of adequate sets in Section 2.3).

A CTL formula is then interpreted with regard to the transition system derived from the reachability graph (see the discussion before Definition 5.19).

The semantics of CTL (see Definition 2.20) has to be slightly adapted to the new atomic propositions, as was the case for LTL formulas. The following semantic definition is almost identical to Definition 5.19 above:

**Definition 5.24** (**Semantic of CTL for safe Eos**). *Let $OS$ be a safe* Eos*, let $p \in prop$ be an atomic proposition, and let $\mu \in R(OS)$ be a node of the reachability graph of $OS$. To define if $OS, \mu \models p$ holds, we distinguish five cases:*

1. *$p = \mathbf{0}$. Then $OS, \mu \models p$ holds if $\mu$ is the empty marking, i.e. if $\mu = \mathbf{0}$.*
2. *$p = \widehat{p_i}[\mathbf{0}]$, $\widehat{p_i} \in \widehat{P}$. Then $OS, \mu \models p$ holds if an empty net token resides on the place $\widehat{p_i}$ in $\mu$*
3. *$p = \widehat{p_i}[*]$, $\widehat{p_i} \in \widehat{P}$ Then $OS, \mu \models p$ holds if a net token resides on the place $\widehat{p_i}$ in $\mu$ that is marked arbitrarily.*
4. *$p = \widehat{p_i}[p_j]$, $\widehat{p_i} \in \widehat{P}$ and $p_j \in P_{d(\widehat{p_i})}$ Then $OS, \mu \models p$ holds if a net token $N$ resides on the place $\widehat{p_i}$ in $\mu$ and in the current marking $M$ of $N$ the place $p_j$ is marked.*
5. *$p = \theta \in \Theta$ Then $OS, \mu \models p$ holds if $\mu \xrightarrow{\theta}$ holds, i.e. if $\theta$ is enabled in $\mu$.*

*The other cases, i.e. the formulas $\neg\phi$, $\phi_1 \wedge \phi_2$, $EX\phi$, $EG\phi$, and $E[\phi_1 U \phi_2]$ are treated exactly as in Definition 2.20.*

*If $\phi$ is a CTL formula we denote by $OS, \mu \models \phi$ that $\phi$ holds in the state $\mu$ of $OS$ according to the inductive definition of $\models$ defined above.*

*$\phi$ is* satisfied *by $OS$ if $OS, \mu_0 \models \phi$ holds. In this case, $OS$ is a* model *for $\phi$ which is also shortly denoted by $OS \models \phi$.*

Note that we say that the Eos satisfies the formula and not only the reachability graph, from which the LTS is actually obtained.

We mention some examples for CTL formulas in the context here: The formula $AG\neg(\widehat{p_1}[*] \wedge \widehat{p_2}[*])$ expresses that in no reachable marking one object net is on place $\widehat{p_1}$ and another on $\widehat{p_2}$, where the object nets' markings are ignored. $EF(\widehat{t}[\vartheta] \wedge \widehat{p_1}[p_4])$ expresses that a computation exists, where the event $\widehat{t}[\vartheta]$ is enabled and the system net place $\widehat{p_1}$ is at the same time marked with an object net, whose place $p_4$ is marked. Note that there may be other tokens in this object net and that on the system net level other object nets may be. If one wants to exclude such things one has to use subformulas of the kind $\neg\widehat{p_1}[p_5]$ to express that the place $p_5$ of the object net, which resides on the system net place $\widehat{p_1}$ is *not* marked – which is also fulfilled, if there is no object net on $\widehat{p_1}$ at all. Liveness of an event $\widehat{t}[\vartheta]$ can be expressed by $AGEF\widehat{t}[\vartheta]$.

In the following we devise a deterministic algorithm that, given a safe(3) or safe(4) Eos $OS$ and a CTL formula $\phi$, decides in space polynomial in $|OS|$ and $|\phi|$, if $OS$ satisfies $\phi$, i.e. if $OS \models \phi$ holds.

In the algorithm we will break up the formula into smaller subformulas and decide if these subformulas hold in certain states. Since one can not check possible infinite paths, the formulas $E[\phi_1 U \phi_2]$ and $EG\phi$ have to be treated carefully. They will be treated in separated subroutines. The main algorithm is stated in Algorithm 3 which is an adaption of the algorithm in [Esp98a] to our case of safe Eos.

If a Eos $OS$ and a CTL formula $\phi$ is given, `check`$(\mu_0, \phi)$ returns `true` if and only if $OS \models \phi$ holds.

We will analyse the exact space requirements of this algorithm later, after we have introduced the subroutines, but note that all calls to subroutines work with shorter formulas.

The syntactic checks, that is which kind of formula $\phi$ presents, are easily possible as is the test if $\phi$ represents a marking or an event in case of $\phi = p \in prop$.

Then, to check the base case, i.e. if $p$ holds in $\mu$ in line 2, it is necessary to check if one of the semantic conditions of Definition 5.24 hold.

**Lemma 5.25.** *Let $OS$ be a safe* Eos*, $\mu$ a marking of $OS$, and $p \in prop$. It is possible to decide in polynomial time in the size of $OS$ if $OS, \mu \models p$ holds.*

*Proof.* Since $\mu$ is given as an array of $|\widehat{P}|$ elements where each element either points to *null*, if the corresponding system net place is unmarked, or to an

**Algorithm 3** `check`$(\mu, \phi)$. Main procedure to check if $\phi$ holds in $\mu$.

```
1: if φ = p ∈ prop then
2:   if p holds in μ then
3:     return true
4:   else
5:     return false
6:   end if
7: else if φ = ¬φ1 then
8:   return not check(μ, φ1)
9: else if φ = φ1 ∧ φ2 then
10:  return (check(μ, φ1) and check(μ, φ2))
11: else if φ = EXφ1 then
12:  for all μ' and τ̂[ϑ] with μ --τ̂[ϑ]--> μ' do
13:    if check(μ', φ1) then
14:      return true
15:    end if
16:  end for
17:  return false
18: else if φ = EGφ1 then
19:  return checkEG(μ, φ1)
20: else if φ = E[φ1 U φ2] then
21:  return checkEU(μ, φ1, φ2)
22: else
23:  return false
24: end if
```

array of $|P|$ elements where $P$ is the set of places of the object net type that resides on that system net place (see Definition 3.3), the first three cases of Definition 5.24 can be easily checked by a couple of table lookups.

For the fourth case, i.e. $p = \theta = \widehat{t}[\vartheta] \in \Theta$ is an event, one at first verifies that all places in the preset of $\widehat{t}$ are marked. Then for each $N \in \mathcal{N}$ it is verified if the transition $\vartheta(N)$ is enabled in the marking that results from summing up all markings of nets of type $N$ in the preset of $\widehat{t}$.

While the fourth case is in general the most time consuming of the four cases, the necessary computations can be done in polynomial time. □

In the case of $\phi = EX\phi_1$ it is also possible to enumerate all markings and events in polynomial space by reusing the same space.

**Lemma 5.26.** *The computations in line 12 of Algorithm 3 can be implemented using polynomial space in the size of OS and $\phi$.*

*Proof.* To store a marking and an event only polynomial space is needed in the size of the input. Moreover, markings and events can be easily enumerated.

For a fixed event it is then easy to verify if it is enabled at the current marking and if firing it results in the currently guessed successor marking $\mu'$. See also Lemma 5.21. □

There are more efficient ways to deal with line 12 of Algorithm 3. Given $\mu$ only a finite number of events are enabled and these can be constructed from $\mu$ using the enabling predicate (Definition 3.7). If $\mu$ and an event are fixed, the possible modes $(\lambda, \rho)$ only differ in the choice of $\rho$ due to the possible distribution of the markings of net tokens of the same type. These can be easily enumerated.

Thus only a counter for the markings and another for the events is needed and altogether in Algorithm 3 it is only necessary to store a constant number of markings, events and (sub-)formulas and then call a subroutine or work recursively on a *smaller formula*.

It remains to devise algorithms for $E[\phi_1 U \phi_2]$ and $EG\phi$, that is the subroutines `checkEG` and `checkEU`. These are again adaptions of the algorithms in [Esp98a] to our case of safe EOS. We start with `checkEU`.

Given a node $v$ and formulas $\phi_1$ and $\phi_2$, it is necessary to check if $v \models E[\phi_1 U \phi_2]$ holds. However, the standard approach in which one deterministically checks all paths does not work here. This was pointed out in [Esp98a]. In short the problem is that a nondeterministic $\Omega(n)$ space algorithm for $E[\phi_1 U \phi_2]$ results in a $\Omega(n^2)$-space deterministic algorithm, but unfortunately the space can not be easily reused and for $E[E \ldots E[\phi_1 U \phi_2] \ldots] U \phi_{m-1}] U \phi_m]$ not less than $\Omega(n^{2m})$ space would be needed, which is exponential in the size of the formula.

We will adapt the algorithm from [Esp98a] to devise a deterministic polynomial space algorithm. In essence the approach works because of the finiteness of the state space and is thus not applicable for safe(1) or safe(2) EOS.

The key is to reduce the possibly infinite number of paths needed to be checked to a finite number by exploiting that only finitely many different markings exists.

For this let $E[\phi_1 U_b \phi_2]$ be a new operator, describing that there exist a computation in which $\phi_1$ is true until $\phi_2$ is true and $\phi_2$ will be true after at most $b$ steps. Formally:

$$OS, \mu \models E[\phi_1 U_b \phi_2] \quad \text{iff} \quad \begin{array}{l} \text{a path } \pi = \mu_1 \mu_2 \ldots \text{ starting at } \mu \text{ exists such that} \\ \text{a } j \text{ with } 1 \leq j \leq b-1 \text{ exists with } OS, \mu_j \models \phi_2 \\ \text{and } OS, \mu_i \models \phi_1 \text{ holds for all } i < j \end{array}$$

We need the following easy to prove lemma.

**Lemma 5.27.** *Let OS be a safe(3) or safe(4)* EOS*, let $k := |\widehat{P}|$ be the number of places of the system net and $l := \max\{|P_N| \mid N \in \mathcal{N}\}$ be the maximum number of places of the object nets. Let $n := \max\{k, l\}$ and let $\mu$ be a node of $RG(OS)$. Then*

$$OS, \mu \models E[\phi_1 U \phi_2] \iff OS, \mu \models E[\phi_1 U_{2^{n^2+n}} \phi_2].$$

**Algorithm 4** `checkEU`$(\mu, \phi_1, \phi_2)$. Check if $\mu \models E[\phi_1 U \phi_2]$ holds.

```
1: for all markings μ' and numbers k with 0 ≤ k < 2^(n^2+n) do
2:    if path(μ, μ', φ1, φ2, k) then
3:       return true
4:    end if
5: end for
6: return false
```

*Proof.* The proof for 1-safe p/t nets can be found in [Esp98a] and can be easily adapted to our case. The main difference is that a 1-safe p/t net with $k$ places may have $2^k$ different markings, whereas a safe(3) or safe(4) Eos may have up to $(1 + 2^l)^k$ different markings (see the proof of Theorem 5.11). With $n \geq k, l$ we have

$$(1 + 2^l)^k \leq (1 + 2^n)^n \leq (2^n + 2^n)^n = (2^{n+1})^n = 2^{n^2+n}$$

resulting in the bound as stated in the lemma above. □

Lemma 5.27 states that if $E[\phi_1 U \phi_2]$ holds for some node $\mu$, then a number $k \leq 2^{n^2+n}$ exists and a path of length $k$ starting at $\mu$ such that $E[\phi_1 U_k \phi_2]$ holds in $\mu$ due to that path. A subroutine `path` is used in the algorithm for $E[\phi_1 U \phi_2]$ utilising this. `path` has five arguments: Two markings $\mu$ and $\mu'$, two formulas $\phi_1$ and $\phi_2$ and a number $k$. `path` returns true if and only if in $RG(OS)$ a path $\mu_0, \mu_1, \ldots, \mu_k$ exists, where $\mu_0 = \mu$, $\mu_k = \mu'$, $\mu_k \models \phi_2$ holds, and for all $i < k$ $\mu_i \models \phi_1$ holds. To decide if $E[\phi_1 U \phi_2]$ holds in a node $\mu$ it thus suffice to call `path`$(\mu, \mu', \phi_1, \phi_2, k)$ for all markings $\mu'$ of $RG(OS)$ and all $k < 2^{n^2+n}$. The algorithm for $E[\phi_1 U \phi_2]$ is stated as Algorithm 4 above. There $n$ is defined as in the statement of Lemma 5.27, i.e. as the maximum over the number of places of the system net and the number of places of all object nets, and is easily obtained once at the beginning.

Each iteration of the loop of Algorithm 4 can reuse the same space. Thus the space needed is the space needed for one execution of `path` plus the space needed to store the marking $\mu'$ and the number $k$ which are both polynomial in the input. Therefore, if `path` can be implemented in polynomial space, so can Algorithm 4.

Since the paths to be checked may have a length of up to $2^{O(n^2)}$, which is exponential in the size of the Eos, a simple brute force or backtracking algorithm does not work here. However, the technique that was used in Savitch's Theorem [Sav70] to prove that PSpace equals NPSpace is applicable: The path is split into two paths of half size each and these path are checked recursively reusing space. This is carried out in Algorithm 5.

To analyse the space complexity of Algorithm 5 first note that the two base cases in line 1 and 4 can be implemented using only polynomial space. For the case $k = 0$ one only needs to check if two markings are equal. The case for

**Algorithm 5** $\texttt{path}(\mu, \mu', \phi_1, \phi_2, k)$

```
1: if k = 0 and μ = μ' and check(μ, φ2) then
2:   return true
3: end if
4: if k = 1 and μ --τ̂[ϑ]--> μ' for some event τ̂[ϑ] then
5:   if check(μ, φ1) and check(μ', φ2) then
6:     return true
7:   end if
8: end if
9: for all markings μ'' do
10:  if path(μ, μ'', φ1, φ1, ⌈k/2⌉) and path(μ'', μ', φ1, φ2, ⌊k/2⌋) then
11:    return true
12:  end if
13: end for
14: return false
```

$k = 1$ in line 4 is similar to the case treated in Lemma 5.26. Here we do not even need to enumerate the markings, but only the events. Again, as discussed after the proof of Lemma 5.26, there are more efficient ways as to enumerate all events, but this suffices here.

To complete the analysis, we first observe that `path` is a recursive procedure whose nesting depth is $\log k$. Furthermore, note that the procedure `check` is called with different markings, but always with $\phi_1$ or $\phi_2$, which in particular are formulas shorter than $E[\phi_1 U \phi_2]$. Moreover, no new formulas need to be stored, since only $\phi_1$ and $\phi_2$ are used. To estimate the space needed by `checkEU`, let $s(\phi)$ be the maximum over all markings $\mu$ of the space required by $\texttt{check}(\mu, \phi)$. Since each call to `path` in `checkEU` may reuse the same space, the nesting depth is at most $\log 2^{n^2+n} = n^2 + n = O(|OS|^2)$, and for each recursive call we only need to store a constant number of markings and integers on the stack, where a marking of a safe Eos has a size quadratic in the size of the Eos and the integers can be encoded using at most $O(n^2)$ bits and are thus quadratic in the size of the Eos, too. It follows that $\texttt{checkEU}(\mu, \phi_1, \phi_2)$ needs space in $O(\max\{s(\phi_1), s(\phi_2)\} + |OS|^2 \cdot |OS|^2)$.

**Lemma 5.28.** *Algorithm 4 including its subroutine in Algorithm 5 can be implemented using polynomial space in the size of OS and $\phi$.*

It remains to develop an algorithm for $EG\phi$. At first, to deal with the possibly infinite number of paths and the possibly infinite length of them an operator $EG_b$ is introduced to reduce this to only finitely many markings that need to be checked. The operator $EG_b$ is similarly defined as $EU_b$ and the corresponding lemma is proven similarly to Lemma 5.27. The operator $EG_b$ is

**Algorithm 6** `checkEG`$(\mu, \phi)$. Check if $\mu \models EG\phi$ holds.

```
1: for all markings μ' do
2:   if path(μ, μ', φ, φ, 2^(n²+n)) then
3:     return true
4:   end if
5: end for
6: return false
```

defined as follows:

$$OS, \mu \models EG_b\phi \quad \text{iff} \quad \begin{array}{l}\text{a path } \pi = \mu_0\mu_2 \ldots, \mu_b \text{ starting at } \mu \text{ exists such that} \\ OS, \mu_i \models \phi \text{ holds for all } 1 \leq i \leq b.\end{array}$$

Similar to Lemma 5.27 we can prove the following lemma:

**Lemma 5.29.** *Let $OS$ be a safe(3) or safe(4)* Eos*, let $k := |\widehat{P}|$ be the number of places of the system net and $l := \max\{|P_N| \mid N \in \mathcal{N}\}$ be the maximum number of places of the object nets. Let $n := \max\{k, l\}$ and let $\mu$ be a node of $RG(OS)$. Then*

$$OS, \mu \models EG\phi \iff OS, \mu \models EG_{2^{n^2+n}}\phi.$$

*Proof.* The direction from left to right is immediate. For the other direction suppose $EG_{2^{n^2+n}}\phi$ holds in a node $\mu$, that is a path $\mu, \mu_1, \ldots, \mu_b$ exists and $\phi$ holds on all nodes of this path. The nodes are actually reachable markings of the Eos $OS$, which has at most $2^{n^2+n}$ different markings. Thus, if all nodes on the path are different, then $\phi$ holds in all nodes and $EG\phi$ holds on all paths. If $\mu_i = \mu_j$ $(i < j)$, then the subpath between $\mu_i$ and $\mu_j$ can be glued after $\mu_j$ infinitely often and we obtain a path that proves that $EG\phi$ also holds in $\mu$. □

Similar to Lemma 5.27 and the resulting Algorithm 4, using Lemma 5.29 Algorithm 6 follows naturally to solve the problem if $\mu \models EG\phi$ holds. Again $n$ is defined as in Lemma 5.29, i.e. the maximum over the number of places of the system net and the number of places of all object nets.

An analysis similar to the one for `checkEU` shows that `checkEG`$(\mu, \phi)$ needs $O(s(\phi) + |OS|^2 \cdot |OS|^2)$ space.

**Lemma 5.30.** *Algorithm 6 including its subroutine in Algorithm 5 can be implemented using polynomial space in the size of $OS$ and $\phi$.*

Putting the algorithms and lemmata together the following theorem can be proven.

**Theorem 5.31.** *Given a safe(3) or safe(4)* Eos *$OS$ and a CTL formula $\phi$ checking whether $OS$ satisfies $\phi$ can be done in $O(|OS|^4 \cdot |\phi|)$ space.*

**Algorithm 7** `IsEOSLive`($OS$). Check if $OS$ is live.

```
1: for all (t̂, t_1, ..., t_|N|) ∈ T̂ × T_1 × ... × T_|N| do
2:   if (t̂, t_1, ..., t_|N|) encodes an event t̂[ϑ] then
3:     if !check(μ_0, AGEF t̂[ϑ]) then
4:       return false
5:     end if
6:   end if
7: end for
8: return true
```

*Proof.* From Lemmata 5.25 and 5.26 dealing with the atomic propositions and the *next*-connective, from Algorithms 4 and 5 and Lemma 5.28 dealing with the *until*-connective, and from Algorithm 6 and Lemma 5.30 dealing with the *global*-connective, it follows that polynomial space in the size $OS$ and the size of $\phi$ suffices to decide deterministically if $OS \models \phi$ holds.

The precise space bound in the statement follows directly from Lemmata 5.28 and 5.30. □

To solve the liveness problem for safe Eos, a last, subtle problem remains. Due to Theorem 5.31 above it is possible to decide if a safe Eos satisfies a CTL formula and due to the definition of the atomic propositions *prop* it is easily possible to express liveness of *one* event as a CTL formula. Unfortunately, exponentially many events in the size of the Eos may exist due to Lemma 3.10 and the discussion preceding it, which is still valid for safe(3) and safe(4) Eos.

Thus simply using the $\wedge$-operator to link the formulas for the liveness of a single event, will result in a formula whose length is exponential in the size of the Eos and while Theorem 5.31 still holds, liveness of safe(3) or safe(4) Eos is not decided in space polynomial *in the size of the* Eos, but in space polynomial in the size of the Eos *and the formula*.

Fortunately, there are two easy ways out of this. An event can be encoded as an array of size $1 + |\mathcal{N}|$ encoding the system net transition and for each $N \in \mathcal{N}$ an object net transition that should fire synchronously. It is thus possible to enumerate all possible combinations from $\widehat{T} \times T_1 \times \ldots \times T_{|\mathcal{N}|}$ check if this is indeed an event by using the labelling functions and, if it is, check check if it is live, whereas the same space is reused for each event. This approach is sketched in Algorithm 7 where the abbreviation $AG\phi \equiv \neg EF\neg\phi$ is used. Algorithm 7 decides the liveness problem in space polynomial in the size of a given safe Eos.

Alternatively, the syntax of CTL formulas can be extended so that formulas like $\bigwedge_{\widehat{\tau}[\vartheta]\in\Theta} AGEF\widehat{\tau}[\vartheta]$ are possible. In this way, the idea present in Algorithm 7 would be incorporated in the CTL model checking algorithms.

Usually, $\Theta$ will not be so large, since in most cases different transitions in the object nets will use different channels. A third alternative is thus to demand

that the labelling is of such a kind that $|\Theta|$ is polynomially bounded in the size of the Eos which is the case by definition for deterministic and strongly deterministic generalised state machines (see Lemma 4.15).

Together with the PSpace-hardness result from Thorem 5.18, the discussion of Algorithm 7 implies the following corollary.

**Corollary 5.32.** *The liveness problem for safe* Eos *is* PSpace*-complete.*

Thus, by Theorem 5.22 and 5.31 given a LTL or CTL formula $\phi$ and a safe Eos $OS$, it can be decided in polynomial space in the size of $\phi$ and $OS$ if $OS \models \phi$ holds. This implies the decidability of many important questions with affordable resources and is a major improvement compared to many of the complexity bounds presented in Chapter 4. The next section deals with the possibility to lower this bound even more in the light of safe Eos with additionally structural restrictions.

## 5.4 Structural Restrictions of Safe Eos and GSMs

The dynamic restriction to safe Eos can be combined with the structural restrictions introduced in Chapter 4. While for p/t net systems this results in better complexity bounds for the reachability problem in many cases, this is often not the case for object net systems. Returning to the construction in the proof of Lemma 4.20 again, where it was proven by simulation of a linear bounded automaton that the reachability problem for strongly deterministic ppGSMs is PSpace-hard, it is observable that the GSM constructed is indeed a strongly deterministic and safe(4) ppGSM. Since ppGSMs are a special case not only of GSMs, but also of free-choice GSMs and since the result also carries over to deterministic GSMs and unrestricted GSMs with respect to determinism, this implies:

**Theorem 5.33.** *The reachability problem for safe ppGSMs, safe free-choice GSMs, and safe GSMs is* PSpace*-complete in the general case as well as in the case of the deterministic or strongly deterministic variants of each of these net classes.*

Moreover, since a GSM is a special kind of a conservative Eos, this result generalises to conservative Eos as well.

**Theorem 5.34.** *The reachability problem for safe conservative* Eos *as well as for the deterministic or strongly deterministic variant is* PSpace*-complete.*

Turning to conflict-free GSMs, the same complexity bound known for safe and conflict-free p/t nets (see Theorem 2.53 and [HR89]) can be established for

Table 5.1: The results obtained so far for *safe* Eos with further structural restrictions.

| | strongly deterministic | deterministic | general |
|---|---|---|---|
| ttGSM | P | PSpace | PSpace |
| ppGSM | PSpace-complete | PSpace-complete | PSpace-complete |
| ptGSM | NP-hard, PSpace | PSpace-complete | PSpace-complete |
| tpGSM | PSpace | PSpace | PSpace |
| acGSM | PSpace | PSpace | PSpace |
| cfGSM | P | PSpace | PSpace |
| fcGSM | PSpace-complete | PSpace-complete | PSpace-complete |
| GSM | PSpace-complete | PSpace-complete | PSpace-complete |
| cEos | PSpace-complete | PSpace-complete | PSpace-complete |
| Eos | PSpace-complete | PSpace-complete | PSpace-complete |

safe, conflict-free, and strongly deterministic GSMs by the technique used in the proof for conflict-free GSMs, i.e. by showing that the reference net inherits properties from the GSM:

**Theorem 5.35.** *Let $G = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be a strongly deterministic, safe and conflict-free GSM. The p/t net* $\text{Rn}(G)$ *is a safe and conflict-free p/t net.*

*Proof.* This can be proven analogously to Theorem 4.35. To prove safeness use Theorem 4.8. □

In [HR89] it is shown that the reachability problem for safe and conflict-free p/t net systems is solvable in polynomial time and since the construction of the reference net in Theorem 5.35 is possible in polynomial time, the reachability problem for safe, conflict-free and strongly deterministic GSMs can be solved quickly:

**Corollary 5.36.** *The reachability problem for safe, conflict-free, and strongly deterministic GSMs is solvable in polynomial time.*

The reachability problem for the other cases, i.e. for safe and conflict-free but not necessarily strongly deterministic GSMs as well as for safe acyclic GSMs and for safe pt-, tp-, and ttGSMs is solvable in PSpace by the general result obtained in this chapter for Eos. However, it is not known if these bounds are optimal. In the case of safe acyclic GSMs the usage of the reference net as in Theorem 5.35 is not possible. The GSM in Figure 4.8 is not only acyclic, but also safe(3). Thus not only for acyclic, but also for safe(3), acyclic, and strongly deterministic GSMs the reference net is not acyclic.

In Table 5.1 the results obtained for safe Eos with further structural restrictions are summarised.

Turning to formalisms that are system safe, it was shown in Theorem 5.12 and Theorem 5.14 that system safeness is not a crucial restriction for Eos or conservative Eos. In both cases the reachability problem remained undecidable.

For system safe GSMs the problem is decidable due to Theorem 4.9, but the problem is ExpSpace-hard. The GSM constructed in Theorem 4.38 to prove that the reachability problem for free-choice GSMs is ExpSpace-hard, is safe(2) and thus the result carries over to system safe free-choice GSMs and system safe GSMs. This even holds if the GSM is strongly deterministic, since the modifications from the proof of Theorem 4.39 are possible here, too.

**Theorem 5.37.** *The reachability problem for system safe free-choice GSMs and system safe GSMs is* ExpSpace*-hard in the general case as well as in the case of the deterministic or strongly deterministic variants of these net classes.*

Furthermore, the hardness results from acyclic as well as for conflict-free p/t nets also carry over to system safe acyclic GSMs resp. system safe conflict-free GSMs, since the p/t net can simply be used as the sole object net in the GSM as was done, for example, in the proof of Theorem 4.34.

The results for strongly deterministic conflict-free GSMs (Theorem 4.35 and Corollary 4.36) and strongly deterministic ttGSMs (Theorem 4.25 and Corollary 4.26) also hold for the system safe variants, because the reference net in this weaker variant is again a conflict-free p/t net resp. a T-net. Thus the reachability problem for system safe, strongly deterministic, and conflict-free GSMs is in NP and the reachability problem for system safe and strongly deterministic ttGSMs is in P.

At last, ppGSMs and ptGSMs are by definition system safe (see Lemma 4.19 and the discussion following it) and thus the results obtained for them in Chapter 4 (see Lemmata 4.20 and 4.21 and Theorems 4.28 and 4.29) also hold for the system safe variants.

In Table 5.2 the results obtained for system safe Eos are summarised. This Table is identical to Table 4.2 summarising the results for Eos with some kind of structural restriction but without a safeness restriction.

Even if in some cases stronger lower bounds might be found for the unsafe versions and better algorithms for the system safe versions, establishing differences between the formalisms with regard to the reachability problem, the lower bounds already proven make it unlikely that the system safe variants are of much use when it comes to verification. The PSpace-results for the safe variants make these far more attractive in this matter. The possibility to decide in PSpace if an Eos is indeed safe, makes this formalism even more attractive.

Table 5.2: The results obtained so far for *system safe* Eos with further structural restrictions.

| | strongly deterministic | deterministic | general |
|---|---|---|---|
| ttGSM | P | ? | ? |
| ppGSM | PSpace-complete | PSpace-complete | PSpace-complete |
| ptGSM | NP-hard | PSpace-hard | PSpace-hard |
| tpGSM | ? | ? | ? |
| acGSM | NP-hard | NP-hard | NP-hard |
| cfGSM | NP-complete | NP-hard | NP-hard |
| fcGSM | ExpSpace-hard | ExpSpace-hard | ExpSpace-hard |
| GSM | ExpSpace-hard | ExpSpace-hard | ExpSpace-hard |
| cEos | undecidable | undecidable | undecidable |
| Eos | undecidable | undecidable | undecidable |

## 5.5 Summary

While the structural restrictions employed to Eos in Chapter 4 lead to decidable formalisms, the reachability problem is still hard to solve for most of them and especially even for some very restricted formalisms like ppGSMs, for example.

In this chapter, we therefore turned to restrictions on the dynamical behaviour of the object net system. We introduced persistent and unary Eos in which conflicts are ruled out dynamically: In the case of persistent Eos by demanding that all transitions that are enabled at a reachable marking indeed can fire and in the case of unary Eos by demanding that at most one event is enabled at each reachable marking, in which case conflicts simply do not exist.

Both formalisms are Turing-complete, a result presented here for the first time.

Motivated by the idea for p/t nets that a safe p/t net's marking can be represented as a set, we then introduced four different kinds of safeness. The restriction to safe(1) and safe(2) Eos results in Turing-complete formalisms, while safe(3) and safe(4) Eos have a finite state space and thus standard problems like reachability and liveness are decidable. Safeness was introduced and investigated in [KBH10b] by me and Michael Köhler-Bußmeier, see also [KBH10a].

We then showed that reachability and liveness can not only be decided for safe(3) Eos, but that these problems can be decided in polynomial space and actually not only them but every property that can be expressed in the temporal logics LTL or CTL, generalising a result by Esparza [Esp98a] for 1-safe p/t nets. These results have been published as joint work with Michael Köhler-Bußmeier in [KBH10b], [KBH10a] and [KBH11b].

The results from Section 5.4 combining structural and dynamic restrictions and presented here for the first time, indicate the importance of safeness, since

relaxing the safeness notion to system safeness results in complexities that are in most cases as bad as in the formalisms presented in the preceding chapter (see Tables 5.1 and 5.2). Furthermore, it is also evident in Section 5.4 that the combination of structural and dynamic restrictions is not worthwhile in many cases, since the bound usually does not drop below PSpace.

Safeness thus gives rise to formalisms that still have the nice modelling features like nesting and mobility of nets as well as synchronisation between them, while, on the other hand, allow to verify if certain properties are fulfilled using arguably adequate resources.

# 6 Object Net Systems

While the Eos introduced in Chapter 3 allow to model mobile objects placed in an environment, they lack certain modelling capabilities. First of all, in some applications it might be desirable not to be restricted to two levels of nesting. In the example in Section 3.1 (see Figures 3.1 and 3.2) the robot picks up an object. In some settings the object itself might be modelled by a Petri net again, in which case a further nesting level is required. Moreover, the object's change of location from, e.g., a desk to the hands of the robot is only modelled by the naming of the places. Here it would be convenient if the object net itself *moves in the vertical dimension*, i.e. the object net residing in the system net at first is moved by firing a synchronous event to the object net modelling the robot. For this the channels would not only be used for synchronisation between object nets but also for the transportation of another object net.

In this chapter, we introduce the formalism of *object net systems* (ONS) that allows exactly this behaviour. For this, the channels, the labelling, and the firing rule have to be adjusted, as well as the definition of markings. An ONS may have an arbitrary nesting depth which might grow or shrink during firing by object nets that travel in the vertical dimension. With these features it is possible to simulate counter machines by encoding the state of a counter by the nesting depth of certain recursively nested object nets. ONS are introduced in Section 6.1 where it is also shown that they are Turing-complete.

In Sections 6.2 and 6.3, we treat restrictions of ONS. By restricting the labelling and channels allowed in an ONS to those that only permit synchronisation and not the transport of tokens in the vertical dimension, a formalism similar to Eos but with a fixed nesting depth naturally arises. In Section 6.2 we introduce this formalism and a notion of safeness similar to the definitions in Chapter 5. We prove that it is still possible to verify properties expressed in the temporal logics LTL or CTL using only polynomial space – albeit the polynomial grows.

In Section 6.3, we introduce a strong variant of safeness for ONS forbidding the creation or destruction of net tokens. This is thus a similar restriction as was imposed for GSMs, but also slightly more liberal, since more than one net token of the same type may be present. In this formalism, the reachability problem, as well as any property expressed in LTL or CTL, can be solved in polynomial space, too.

Object net systems with channels that allow the vertical transport of tokens have been introduced in [KBH09] by Michael Köhler-Bußmeier and myself.

The formalisms there is, however, rather complicated, mainly due to the firing rule in which whole trees of transitions are allowed to fire. A simpler version was therefore introduced in [HKB12a] by me and Michael Köhler-Bußmeier. This simpler version is presented here. It still captures the essentials of the formalism and in particular still allows the vertical transport of tokens, but firing is restricted to two adjacent nesting levels.

Apart from a proof sketch in [HKB12a] for Theorem 6.22 below, the work concerning the restriction of ONS to a fixed depth and the results concerning the safe variants are presented here for the first time.

## 6.1 Fundamentals of ONS

In the following we introduce Object Net Systems (ONS) as a generalisation of Eos. ONS have an arbitrary nesting depth and 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. For example one could model the different objects on places $\widehat{p}_A$, $\widehat{p}_B$ and so on in Figure 3.1 by object nets if these objects have inner behaviour or allow interaction with them. By using a different kind of channel the event $\widehat{t}_A[t_A]$ could then synchronise the system net and the object net and additionally transport the object net from the place $\widehat{p}_A$ in the system net to the place $p_A$ in the object net. The partial marking $\widehat{p}_A[M] + \widehat{p}_0$ would then have changed to $\widehat{p}_0[p_A[M]]$ where $M$ is the marking of the object net residing on $\widehat{p}_A$ at the beginning. Note that the reached marking has now a nesting depth of 3.

In the following, we introduce the formalism and in particular present the firing rule which is designed in such a way that firing is restricted to at most two adjacent levels of nesting, i.e. synchronisation is only possible with an object net below or above oneself not both and not in more depth.

After introducing the formalism, we prove that in their general form ONS are Turing-complete.

Before introducing the formalism we have a look at another example. In Figure 6.3 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. Channel properties 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 $c$). Ignoring the inner structure of the net tokens both transitions are activated and may fire. The successor marking is pictured in Figure 6.4. 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 a formal description of this formalism is given.

As in the case of Eos 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 the special object net $N_\bullet$, which has no places and no transitions. We set $\widehat{\mathcal{N}} := \mathcal{N} \uplus \{\widehat{N}\}$. Instead of $P_i$, $T_i$ and so on we sometimes make use 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 $\mathcal{P}_\mathcal{N} := \uplus_{N \in \mathcal{N}} P(N)$, $\mathcal{P} := \mathcal{P}_\mathcal{N} \uplus \widehat{P}$, $\mathcal{T}_\mathcal{N} := \uplus_{N \in \mathcal{N}} T(N)$, and $\mathcal{T} := \mathcal{T}_\mathcal{N} \uplus \widehat{T}$ to denote the set of all places and transitions.

The places are all typed via the *typing function* $d : \mathcal{P} \to \mathcal{N}$. 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 an 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 $\epsilon$. The intended meaning of $l(t) = (c, N)$ is that $t$ synchronises 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 labelings together with their intended meaning and the restrictions the labelings impose on the nets' structure.

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

2. $l(t) = (\Uparrow c, N')$, $t \in T(N), N \neq \widehat{N}$. This labelling – that can not be used in the system net $\widehat{N}$ – means that $t$ wants to synchronise (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 6.1 and 6.2). Formally, we demand a place $p'$ in $N'$ with $d(p') = N$ and a transition $t' \in p'^\bullet$ with $l(t') = (\Downarrow c, N)$.

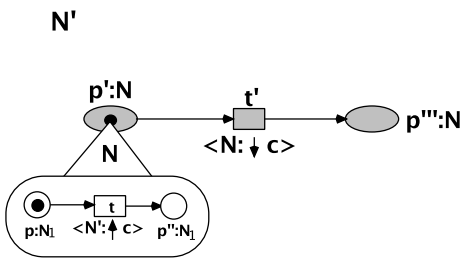


Figure 6.1: Before firing.

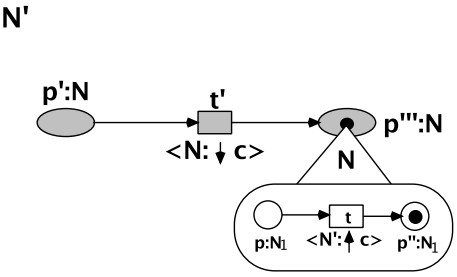


Figure 6.2: After firing $(t, t')$ from Figure 6.1.

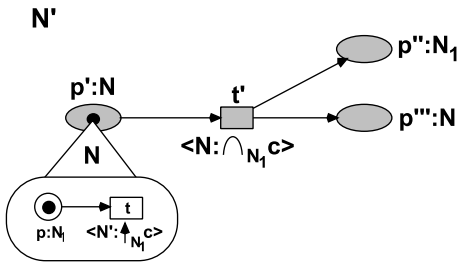


Figure 6.3: Before firing.

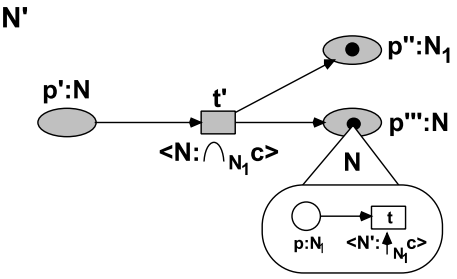


Figure 6.4: After firing $(t, t')$ from Figure 6.3.

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 sent 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 6.3 and 6.4. Note that the token on place $p$ (Fig. 6.3) resp. place $p''$ (Fig. 6.4) can be an object net. Formally there is a place $p'$ with $d(p') = N$ and a transition $t' \in p'^\bullet$ with $l(t') = (\cap_{N_1} c, N)$. Here the usage of the symbol $\cap$ shall illustrate that a net coming from below is "caught". 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 $d(p) = N_1$

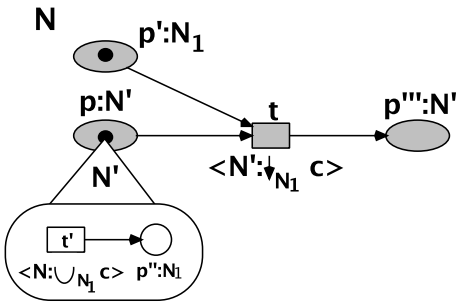


Figure 6.5: Before firing.

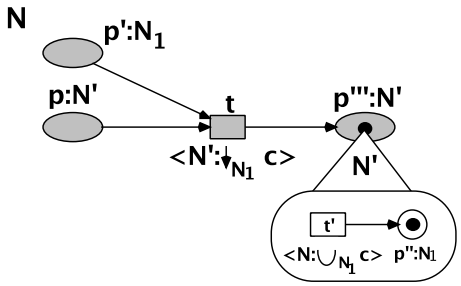


Figure 6.6: After firing $(t, t')$ from Figure 6.5.

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 6.5 and 6.6. Note again that the token on place $p$ (Fig. 6.5) resp. place $p''$ (Fig. 6.6) 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 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, 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. Now let

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

for a $N \in \widehat{\mathcal{N}}$ and let $\mathcal{M}_0 := \cup_{N \in \widehat{\mathcal{N}}} \mathcal{M}_0(N)$. A multiset $\mu \in \mathcal{MS}(\mathcal{M}_0(N))$ for a fixed $N \in \widehat{\mathcal{N}}$ describes how many *empty* net tokens reside on each place of $N$, including black tokens, which are just special net tokens. The multiset $\mu$ is thus similar to the usual multiset of places that describes a marking of a p/t net.

Let for each $N \in \widehat{\mathcal{N}}$

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

Finally, let

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

The multisets in these definitions are all finite.

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.

As in Definition 3.2, $|\mu|$ denotes the number of net tokens present in $\mu$ at the top most level, i.e. if $\mu \in \mathcal{MS}(\mathcal{M}_i(N))$ for some $N \in \widehat{\mathcal{N}}$ and $i \in \mathbb{N}$, then

$$|\mu| := \sum_{(p,\mu') \in P(N) \times \cup_{k \leq i} \mathcal{MS}(\mathcal{M}_k(d(p)))} \mu(p, \mu')$$

Also as in Definition 3.2, we define the partial order $\leq$ by setting $\mu_1 \leq \mu_2$ iff $|\mu_1| \leq |\mu_2|$ and the elements of $\mu_1$ and $\mu_2$ can be arranged in such a way that $\mu_1 = \sum_i \widehat{p}_i[M_i]$, $\mu_2 = \sum_j \widehat{p}'_j[M'_j]$ and for all $k \leq |\mu_1|$ we have $\widehat{p}_k = \widehat{p}'_k$ and $M_k \leq M'_k$, where $\leq$ is the recursive application of this relation. The recursion ends with $\leq$ as the usual multiset relation (see Definition 3.2 and also Chapter 2.1). Thus the same net tokens appear in $\mu_2$ with at least the same marking as in $\mu_1$ and in $\mu_2$ additional net tokens may appear.

With $\preceq$ we denote the special partial order with $\mu_1 \preceq \mu_2$ iff $|\mu_1| \leq |\mu_2|$ and the elements of $\mu_1$ and $\mu_2$ can be arranged in such a way that $\mu_1 = \sum_i \widehat{p}_i[M_i]$, $\mu_2 = \sum_j \widehat{p}'_j[M'_j]$ and for all $k \leq |\mu_1|$ we have $\widehat{p}_k = \widehat{p}'_k$ and $M_k = M'_k$. Thus every net token that appears in $\mu_1$ also appears in $\mu_2$ and with the same marking. In $\mu_2$ additional net tokens may appear. This definition is identical to the definition of $\preceq$ for markings of Eos, see Definition 3.2.

For ONS a relation to address the nesting of markings is furthermore needed. We write $\mu \triangledown \mu'$ to indicate that the submarking $\mu'$ is contained in the marking $\mu$ within exactly one level of nesting:

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

The reflexive and transitive closure of this relation is denoted by $\triangledown^*$ as usual. We also use the sets $\triangledown(\mu) := \{\mu' \mid \mu \triangledown \mu'\}$ and $\triangledown(\mu)^* := \{\mu' \mid \mu \triangledown^* \mu'\}$. Thus

$\mu \nabla^* \mu'$ means that $\mu$ contains $\mu'$ at some nesting level which is also expressed by $\mu' \in \nabla(\mu)^*$.

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 $\mu$ 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.

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

where $\mathbf{1}_N : \mathcal{P} \to \{0,1\}$ with $\mathbf{1}_N(p) = 1$ iff $d(p) = N$. Note that the summation in $\Pi^2_N$ 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 $\Pi^2_N$ is useful to describe the firing rule.

### Object Net Systems, Events, and the Firing Rule.

**Definition 6.1** (**Object Net Systems**). *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.*

*Each type appears at most once in the preset and in the postset of a transition, i.e.*

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

*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 $\mu_0 \in \mathcal{MS}(\mathcal{M}_k(\widehat{N}))$.*

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 $l(t') = (\cap_{N_1} c, N)$ (cf. Figures 6.3 and 6.4).

2. $(t, t') \in \mathcal{T} \times \mathcal{T}$ with $l(t) = (\Downarrow_{N_1} c, N')$ and $l(t') = (\cup_{N_1} c, N)$ (cf. Figures 6.5 and 6.6).

3. $(t, t') \in \mathcal{T} \times \mathcal{T}$ with $l(t) = (\Uparrow c, N')$ and $l(t') = (\Downarrow c, N)$ (cf. Figures 6.1 and 6.2).

4. $t \in \mathcal{T}$ with $l(t) = \epsilon$.

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

For the first case, let $\mu$ be the current marking and let $\lambda \preceq \mu'$ in which $\mu \nabla^* \mu'$, i.e. $\mu$ has a nested net token with marking $\mu'$ and $\lambda$ is a submarking of $\mu'$. Furthermore, let $\lambda' \preceq \lambda''$ where $\lambda \nabla \lambda''$. The intended meaning is that $\lambda$ is the sub-marking of $\mu$ enabling $t'$, and $\lambda'$ is the sub-marking (of $\lambda$) that enables $t$ in the synchronous event. Let furthermore $\rho$ and $\rho'$ be two markings with $\rho' \preceq \rho''$, where $\rho \nabla \rho''$. The intended meaning here is that $\rho$ is the resulting sub-marking with regard to $t'$, and $\rho'$ with regard to $t$. Additionally, a net of type $N_1$ is removed from $\lambda'$ and added to $\rho$.

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

$$
\begin{aligned}
\phi_{\Uparrow_{N_1}, \cap_{N_1}}(t, t', \lambda, \lambda', \rho, \rho') \quad \Longleftrightarrow \\
& \Pi^1(\lambda) = \mathbf{pre}(t') \wedge \Pi^1(\rho) = \mathbf{post}(t') \wedge \\
& \Pi^1(\lambda') = \mathbf{pre}(t) \wedge \Pi^1(\rho') = \mathbf{post}(t) \wedge \\
& \forall N' \in \mathcal{N} \setminus \{N, N_1\} : \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) \wedge \\
& \forall N' \in \mathcal{N} \setminus \{N_1\} : \Pi^2_{N'}(\rho') = \Pi^2_{N'}(\lambda') \wedge \\
& \Pi^2_N(\rho) = \Pi^2_N(\lambda) - \lambda' + \rho' \wedge \\
& \Pi^2_{N_1}(\rho) = \Pi^2_{N_1}(\lambda') \wedge \\
& \Pi^2_{N_1}(\rho') = \mathbf{0}
\end{aligned}
\tag{6.1}
$$

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. 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 of the above first case. In the third case no object net is transported and in the fourth case there is not even a synchronisation happening.

For the second case, the firing predicate is given by

$$\begin{aligned}\phi_{\Downarrow_{N_1},\cup_{N_1}}(t,t',\lambda,\lambda',\rho,\rho') \quad \Longleftrightarrow \\ \Pi^1(\lambda) = \mathbf{pre}(t) \wedge \Pi^1(\rho) = \mathbf{post}(t) \wedge \\ \Pi^1(\lambda') = \mathbf{pre}(t') \wedge \Pi^1(\rho') = \mathbf{post}(t') \wedge \\ \forall N' \in \mathcal{N} \setminus \{N, N_1\} : \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) \wedge \\ \forall N' \in \mathcal{N} \setminus \{N_1\} : \Pi^2_{N'}(\rho') = \Pi^2_{N'}(\lambda') \wedge \\ \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) - \lambda' + \rho' \wedge \\ \Pi^2_{N_1}(\rho) = \mathbf{0} \wedge \\ \Pi^2_{N_1}(\rho') = \Pi^2_{N_1}(\lambda)\end{aligned} \tag{6.2}$$

Note that $\lambda$ now enables $t$, $\lambda'$ enables $t'$, $\rho$ is the resulting sub-marking with regard to $t$, and $\rho'$ with regard to $t'$. Furthermore, a net of type $N_1$ is removed from $\lambda$ and added to $\rho'$ (and also to $\rho$, since $\rho' \preceq \rho''$ and $\rho \nabla \rho''$). 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:

$$\begin{aligned}\phi_{\Uparrow,\Downarrow}(t,t',\lambda,\lambda',\rho,\rho') \quad \Longleftrightarrow \\ \Pi^1(\lambda) = \mathbf{pre}(t') \wedge \Pi^1(\rho) = \mathbf{post}(t') \wedge \\ \Pi^1(\lambda') = \mathbf{pre}(t) \wedge \Pi^1(\rho') = \mathbf{post}(t) \wedge \\ \forall N \in \mathcal{N} \setminus \{N'\} : \Pi^2_N(\rho) = \Pi^2_N(\lambda) \wedge \\ \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) - \lambda' + \rho'\end{aligned} \tag{6.3}$$

$$\begin{aligned}\phi_\epsilon(t,\lambda,\rho) \quad \Longleftrightarrow \\ \Pi^1(\lambda) = \mathbf{pre}(t) \wedge \Pi^1(\rho) = \mathbf{post}(t) \wedge \\ \forall N \in \mathcal{N} : \Pi^2_N(\rho) = \Pi^2_N(\lambda)\end{aligned} \tag{6.4}$$

Note that the four firing predicates might at 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 [KBH09] where a tree of synchronous transitions was able to fire. The firing rule can now be stated as follows:

**Definition 6.2** (**Firing Rule**). *Let OS be an ONS and $\mu, \mu' \in \mathcal{M}$ markings. The synchronous event $(t, t')$ is* enabled in $\mu$ for the mode $(\lambda, \lambda', \rho, \rho') \in \mathcal{M}^4$ *iff a $\mu'$ exists with $\mu \nabla^* \mu'$ and $\lambda \preceq \mu'$, a $\lambda_i$ exists with $\lambda \nabla \lambda_i$ and $\lambda' \preceq \lambda_i$, and a $\rho_i$ exists with $\rho \nabla \rho_i$ and $\rho' \preceq \rho_i$, 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 $\mu$ for the mode $(\lambda, \rho) \in \mathcal{M}^2$ iff a $\mu'$ exists with $\mu \nabla^* \mu'$ and $\lambda \preceq \mu'$ and $\phi_\epsilon$ holds for $(t, \lambda, \rho)$.*

*An event $\vartheta$ that is enabled in $\mu$ for a mode can fire: $\mu \xrightarrow[OS]{\vartheta} \mu_{succ}$. The resulting successor marking is defined by removing a nested multiset and replacing it with another one. Let in the definition of $\mu'$ above $k$ be such that $\mu \nabla^k \mu'$, then there*

*are $k$ places $p_1, p_2, \ldots, p_k \in \mathcal{P}$ and $k$ markings $\mu_1, \mu_2, \ldots, \mu_k$ such that $\mu_k = \mu'$ and $\mu(p_1[p_2[\ldots[p_k[\mu_k] + \mu_{k-1}] + \ldots] + \mu_1]) = 1$. The successor marking is then defined by*

$$\begin{aligned} \mu_{succ} = \mu \quad &- \quad p_1[p_2[\ldots[p_k[\mu'] + \mu_{k-1}] + \ldots] + \mu_1] \\ &+ \quad p_1[p_2[\ldots[p_k[\mu' - \lambda + \rho] + \mu_{k-1}] + \ldots] + \mu_1]. \end{aligned}$$

*The nested marking $\mu'$ is thus replaced by $\mu' - \lambda + \rho$. If firing happens at the uppermost level, i.e. in case of $k = 0$, $\mu' \preceq \mu$ holds.*

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

As was the case for Eos, the event $\vartheta$ itself does not fully characterize the firing. For example, a synchronous event $(t, t') \in \mathcal{T} \times \mathcal{T}$ might be possible in different nesting levels. Thus the mode is also important to fully characterize the firing.

### Turing-Completeness of Object Net Systems

Since an Eos can be seen as a special ONS, which does not use the channel properties for the vertical transport of net tokens and also only has two nesting levels, ONS are Turing-complete by Theorem 3.13. In the following we give an alternative proof, to demonstrate the possibilities of object net systems. The ONS constructed in the proof exploits the possibility to transport net tokens vertically to encode counters of a counter program.

**Theorem 6.3.** *Each counter program can be bisimulated by an object net system.*

*Proof.* A counter program (see Definition 2.3) has access to a fixed number $r \geq 2$ of counters and consists of a finite sequence of $m$ commands each of which is either an increase or a decrease operation, an operation that tests if a certain counter is zero or not, jumping accordingly, or a halt-command.

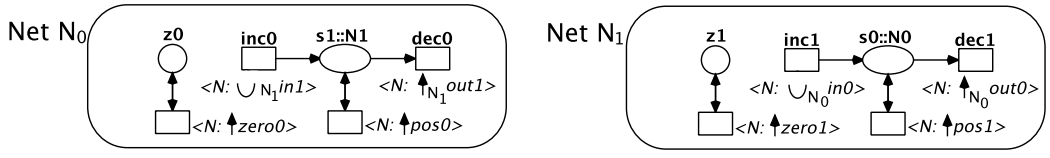


Figure 6.7: The object nets $N_0$ and $N_1$ (from Theorem 6.3).

The counters are encoded by the nesting depth of the two object nets $N_0$ and $N_1$ depicted in Figure 6.7. The structure of both nets is identical, but the

channels used are different. Furthermore, the places $z_0$ and $z_1$ are typed with $N_\bullet$ and $d(s_1) = N_1$ and $d(s_0) = N_0$.

To encode the number 0, the place $z_0$ is marked in the net $N_0$. The number $n+1$ is encoded by a net token of type $N_b$ with $b = ((n+1) \bmod 2)$ whose place $s_{b'}$ with $b' = (n \bmod 2)$ is marked by a net token encoding the number $n$:

$$\begin{aligned} \#0 &:= z_0[\mathbf{0}], \\ &\quad \text{marking of a net of type } N_0 \\ \#(n+1) &:= s_b[N_b, \#n], b = n \bmod 2, \\ &\quad \text{marking of a net of type } (n+1) \bmod 2 \end{aligned}$$

The nets $N_0$ and $N_1$ thus alternate:

$$\#0 = z_0[\mathbf{0}], \quad \#1 = s_0[N_0, z_0[\mathbf{0}]], \quad \#2 = s_1[N_1, s_0[N_0, z_0[\mathbf{0}]]], \quad \ldots$$

Each command $cmd_k$ in the counter program is simulated by a net fragment $\widehat{N}(k)$ depicted in Figure 6.8. The system net $\widehat{N}$ is the union of these $\widehat{N}(k)$ together with the fragment *init* in Figure 6.8.

Some net fragments share places: The places 1, 2, ..., $k$, ..., $n$ encode the current position in the counter program and are typed with $N_\bullet$. Each counter $c_j$ is encoded by the places $p_j^0$ and $p_j^1$ where $d(p_j^0) = N_0$ and $d(p_j^1) = N_1$. For each $j$ only one of $p_j^0$ and $p_j^1$ is marked and the net token residing there will encode the value of the counter as described above.

The net fragment for an increase operation has the two auxiliary places $q_k^0$ and $q_k^1$ and the net fragment for the a decrease operation has the two auxiliary places $gbg_k^0$ and $gbg_k^1$. The places are typed according to their superscripts: $d(q_k^0) = d(gbg_k^0) = N_0$ and $d(q_k^1) = d(gbg_k^1) = N_1$.

Finally, each program configuration $C = (k, c_1, \ldots, c_r)$ is encoded by a marking $\mu(C)$ where

1. From the places $1, 2, \ldots, n$ exactly the place $k$ is marked with a black token.

2. For each counter $c_j$ exactly one of the places $p_j^0$ and $p_j^1$ is marked according to the value of the counter. $p_j^0$ is marked if the value is an even number and $p_j^1$ if the value is an odd number. The places are marked with the net token representing that number. Thus the place $p_j^{c_j \bmod 2}$ is marked with the net token $[N_{c_j \bmod 2}, \#c_j]$ and the place $p_j^{(c_j+1) \bmod 2}$ is unmarked.

3. All other places are unmarked.

Given a counter program $CP$ the object net system $ONS(CP)$ is constructed as described above. The initial marking is $\mu_0 := \mu(C_0)$ where $C_0$ is the initial configuration of $CP$.

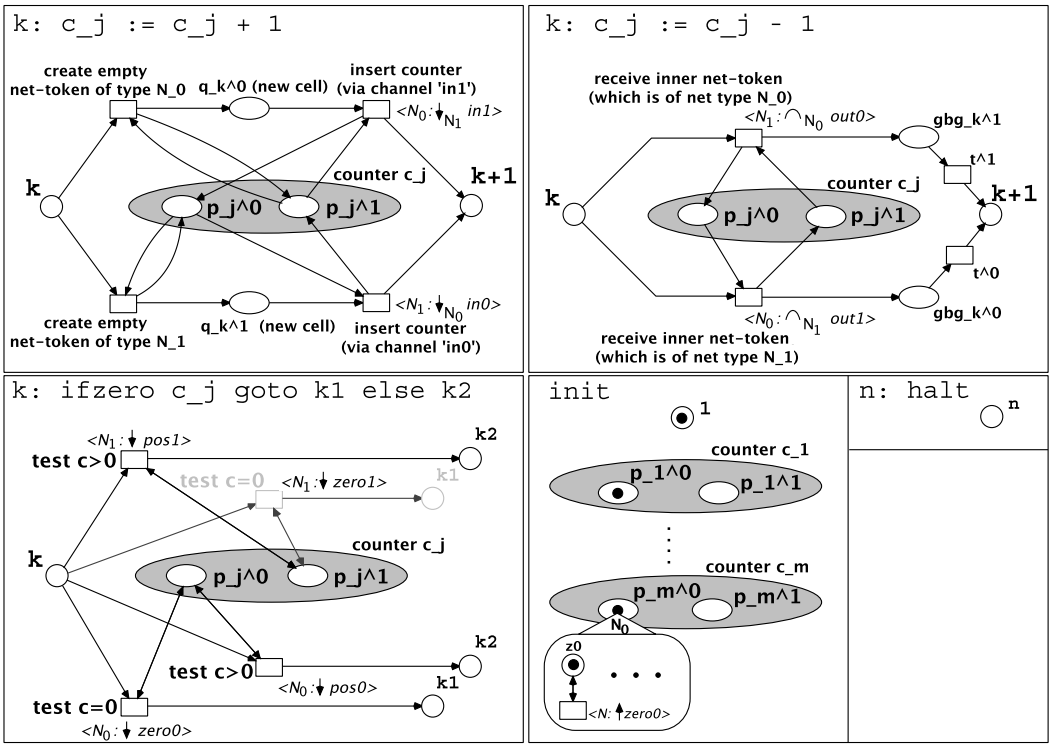


Figure 6.8: Net fragments for the simulation of counter programs (from Theorem 6.3).

We now prove that $ONS(CP)$ simulates $CP$ and vice versa:

$$C_0 \xrightarrow[CP]{*} C \Leftrightarrow \mu(C_0) \xrightarrow[ONS(CP)]{*} \mu C$$

The proof is via induction over the computation length.

In case of computation of length 0, $C = C_0$ holds, which is correctly simulated by $\mu_0 = \mu(C_0) = \mu(C)$.

Let $C_0 \xrightarrow[CP]{*} C \xrightarrow[CP]{} C'$ with $C = (k, c_1, \ldots, c_r)$ and $C' = (k', c'_1, \ldots, c'_r)$. Per assumption $\mu(C_0) \xrightarrow[ONS(CP)]{*} \mu(C)$ holds. In $\mu(C)$ one of the places $1, 2, \ldots, m$ is marked. Let this place be $k$. Then only in the net fragment $\widehat{N}(k)$ events are enabled.

There are now four possible cases depending on the command $cmd_k$ in the counter program:

1. $c_j := c_j + 1$. Depending on the current value of the counter $c_j$ the upper left or lower left transition is enabled. If $c_j$ has an odd value (and thus the place $p_j^1$ is marked) a new net token of type $N_0$ is created on $q_k^0$. Next, the system net and this newly created net token synchronise via

the channel $in_1$, which puts the net token currently on $p_j^1$ onto the place $s_1$ of the newly created net token, which is then placed on $p_j^0$. This correctly encodes the counter's new value and since the place $k+1$ is now marked, the reached marking is $\mu(C')$. The case if $c_j$ has an even value is treated analogously.

2. $c_j := c_j - 1$. Again depending on the current value of the counter $c_j$ the upper or lower transition is enabled. If $c_j$ has an odd value, the system net and the object net on $p_j^1$ synchronise via the channel $out_0$. This puts the net token on place $s_0$ of the object net on $p_j^1$ onto the place $p_j^0$ and the now empty net token of type $N_1$ onto the garbage place $gbg_k^1$ and thus decreases the counter's value by 1. The empty net token on $gbg_k^1$ activates the transition $t^1$ which is the only enabled transition right now. Firing $t^1$ places a token on $k+1$ and the reached marking is then $\mu(C')$. The case if $c_j$ has an even value is again treated analogously.

3. `ifzero` $c_j$ `jump` $k_1$ `else jump` $k_2$. If $p_j^0$ is marked, the net token encodes an even number and possibly 0. If and only if the net token on $p_j^0$ encodes 0, the place $z_0$ of $N_0$ is marked. Otherwise the place $z_0$ is not marked and the place $s_1$ is marked with a net of type $N_1$, which encodes a number greater than 0. In the first case, synchronisation via the channel $zero_0$ is possible, in the second case synchronisation via the channel $pos_0$ is possible. Only one of these two possible events is enabled, depending on the marking, thus the correct place $k_1$ or $k_2$ is marked, arriving at the correct marking $\mu(C')$. If the counter currently has an odd value and thus the place $p_j^1$ is marked, the counter's value is greater than 0 and the place $k_2$ is correctly marked. It is actually not necessary to use the channel $pos_1$ in this case. This is only done for reasons of symmetry. Also the greyed transition and places are not needed because this zero test can never succeed.

4. `halt`. This command terminates the counter program and is also a deadlock in the object net system.

The converse direction is proven analogously, i.e. whenever a marking $\mu(C)$ for some configuration $C$ has been reached and $\mu(C)$ is not a deadlock, it is possible to reach a marking $\mu(C')$ for some configuration $C'$ and if $\mu(C')$ is the first marking which corresponds to a configuration, then $C \xrightarrow[CP]{} C'$ holds.

A increment or decrement command is simulated by two successive events where the second event is the only activated event after firing the first. Starting from $\mu(C_0)$ only few markings are thus reachable that do *not* correspond to a configuration in the counter program and in these configurations exactly one event is enabled. Firing this event then leads immediately to a marking of $ONS(CP)$ that again corresponds to a configuration of $CP$.

Furthermore, the events for decrementing a counter are only possible if the current value of the counter is greater than 0. The test for zero is simulated by three events, but depending on the value of the counter only one of these is enabled. Thus, altogether at most one event is enabled at each reachable marking and this, together with a second event in case of the increase or decrease operation, lead to markings that correspond to configurations in the counter program.

The ONS thus simulates the counter program correctly. □

**Corollary 6.4.** *The reachability problem is undecidable for ONS.*

## 6.2 Eos and GSM with Fixed Nesting Depth

In this section, we introduce Eos and GSM without the restriction of the nesting depth to 2, but with a fixed nesting depth of $k \geq 1$. Albeit a generalisation of Eos as introduced in Chapter 3, they are introduced here as a restriction of object net systems, because the firing rule of object net systems can be easily restricted to fit to the case of constant depth Eos and GSMs.

Hardness and undecidability results carry over to this new class of object systems; for example, Eos with nesting depth $k \geq 2$ are Turing-complete.

However, introducing similar safeness definitions as in Chapter 5, we show that the reachability problem for these safe variants remains in PSpace and also discuss the LTL and CTL model checking problems for these formalisms as we did for safe Eos.

While arbitrary nesting depths have been investigated before in [KR03a] and [KR04], the restriction to a nesting depth with a fixed bound is new, as are the application of safeness and the resulting complexity theoretic results.

### Fundamentals

To restrict the nesting depth of ONS to a constant, the channel properties have to be restricted to $\{\Uparrow, \Downarrow\}$ to prevent a vertical transport of tokens and also to prevent an increase or decrease of the nesting depth during run time. However, an unbounded nesting depth is still possible, for example, by a net of type $N_1$ consisting of a transition $t$ with empty preset and a single place $p$ in the postset of $t$ that again is typed with $N_1$.

Therefore, the idea of Eos to have at one level the system net, at the next level the object nets, and at the last level the black tokens, is generalised. At the uppermost level is still the system net and at the lowermost level are still the black tokens, but the levels in between are defined more gradually.

**Definition 6.5** (**$k$-level Eos and GSMs**). *Let $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ be an object net system with initial marking $\mu_0$. Different from ONS as defined in*

*Definition 6.1 it is not demanded here that each type appears at most once in the preset and in the postset of a transition.*

*OS is an* Eos *with constant nesting depth $k \geq 1$ or a $k$-level* Eos*, denoted by* $\text{Eos}_k$ *for short, if $\widehat{\mathcal{N}} := \{\widehat{N}\} \cup \mathcal{N}$ can be partitioned into $k+1$ pairwise disjoint sets $\mathcal{N}_0, \mathcal{N}_1, \ldots, \mathcal{N}_k$ such that $\mathcal{N}_0 = \{\widehat{N}\}$, $\mathcal{N}_k = \{N_\bullet\}$, and all places of a net $N \in \mathcal{N}_i$, $0 \leq i < k$ are typed only with nets $N' \in \mathcal{N}_j$, where $i < j \leq k$, i.e. $d(\cup_{N \in \mathcal{N}_i} P(N)) \subseteq \cup_{j=i+1}^{k} \mathcal{N}_j$. Furthermore, the channel properties are limited to $\{\Uparrow, \Downarrow\}$.*

*OS is a GSM with constant nesting depth $k \geq 1$ or a $k$-level GSM, denoted by $\text{GSM}_k$ for short, if it is an $\text{Eos}_k$ and additionally for all $N \in \widehat{\mathcal{N}} \setminus \{N_\bullet\}$ the GSM-condition*

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

*holds and each $N \in \widehat{\mathcal{N}} \setminus \{N_\bullet\}$ appears in $\mu_0$ at most once.*

A p/t net $N$ is thus a special $k$-level Eos, where $k = 1$, $\mathcal{N}_0 = \{N\}$, and $\mathcal{N}_1 = \{N_\bullet\}$. Also an Eos $OS = (\widehat{N}, \mathcal{N}, d, l)$ as defined in Chapter 3 is a special $k$-level Eos, where $k = 2$, $\mathcal{N}_0 = \{\widehat{N}\}$, $\mathcal{N}_1 = \mathcal{N} \setminus \{N_\bullet\}$, and $\mathcal{N}_2 = \{N_\bullet\}$.

Since additional levels need not be used, an $\text{Eos}_k$ is also an $\text{Eos}_{k+j}$ for all $j \geq 1$. An analogous statement also holds for $\text{GSM}_k$. The following Lemma follows easily from the definitions above.

**Lemma 6.6.** *An $\text{Eos}_k$ is also an $\text{Eos}_{k+j}$ for all $j \geq 1$. A $\text{GSM}_k$ is also a $\text{GSM}_{k+j}$ for all $j \geq 1$ and also an $\text{Eos}_{k+j}$ for all $j \geq 0$.*

Since $\text{Eos}_2$ are Turing-complete by Theorem 3.13 this result carries over to $k$-level Eos with $k > 2$ due to the above lemma.

**Corollary 6.7.** $\text{Eos}_k$ *are Turing-complete for $k \geq 2$.*

## Safeness for $\text{Eos}_k$

Similar to Eos a safeness notion can be introduced for $\text{Eos}_k$. Here we state a definition that generalises the definition of safe(3) Eos and in particular guarantees a finite state space. We then prove that for $\text{Eos}_k$ with this property the technique applied in the case of safe(3) Eos works, too, and conclude that LTL and CTL model checking is still possible in polynomial space. Due to Lemma 6.6 this results carry over to $\text{GSM}_k$ as well.

**Definition 6.8 (Safeness for $\text{Eos}_k$).** *Let OS be an $\text{Eos}_k$ for some fixed $k \geq 1$. OS is* safe *iff for all reachable markings there is at most one token on each place regardless of the nesting depth in which the place may reside:*

$$\forall \mu \in R(OS, \mu_0) : \forall \mu' \in \nabla(\mu)^* : \forall p \in \mathcal{P} : \Pi^1(\mu')(p) \leq 1$$

Note that Definition 6.8 generalises the definition of safe(3) Eos and safe p/t nets, i.e. a safe $\text{Eos}_2$ is a safe(3) Eos and a safe $\text{Eos}_1$ is a safe p/t net.

Thus in these cases it is immediately clear that the state space is finite. Indeed, this is always the case for a safe $\text{Eos}_k$.

**Theorem 6.9.** *The set of reachable markings of a safe* $\text{Eos}_k$ *is finite.*

*Proof.* Every marking of a safe $\text{Eos}_k$ can be rewritten as a sum of lists of up to $k$ places. For example, let $\mu = p_1[p_1'[\mathbf{0}] + p_3'[p_1'' + p_2'']] + p_2[\mathbf{0}]$. The marking $\mu$ denotes that on the uppermost level the places $p_1$ and $p_2$ are marked. On $p_2$ resides an empty net token and on $p_1$ resides a net token whose place $p_1'$ is marked by an empty net token and whose place $p_3'$ is again marked by a net token whose places $p_1''$ and $p_2''$ are marked. The marking $\mu$ can be rewritten as $p_1[p_1'[\mathbf{0}]] + p_1[p_3'[p_1'']] + p_1[p_3'[p_2'']] + p_2[\mathbf{0}]$. This is possible for every marking $\mu$ and $\mu$ can also be reconstructed as long as the marking is interpreted as a marking of a safe $\text{Eos}_k$.

Now let $n := \max\{|P_N| \mid N \in \widehat{\mathcal{N}}\}$. Then there are $(n+1)^k$ different possibilities to create a list of $k$ places where additionally $\mathbf{0}$ is allowed, too. (Some of these lists are not sensible, for example, $p_1[\mathbf{0}[p_2]]$ is not a marking, but since we are only interested in upper bounds, this is of no concern here.) Each of these lists can appear in a marking or not and thus the number of markings is bounded by $2^{(n+1)^k} = 2^{O(n^k)}$ and is therefore finite. □

The proof of the above theorem is similar to the proof of Theorem 5.11 establishing the finiteness of the state space for safe Eos, which is now also implied by Theorem 6.9 above.

While finite, the bound increases considerably in comparison to safe Eos. For small $k$ the number $2^{(n+1)^k}$ already becomes huge. Nonetheless, since $k$ is fixed, the exponent is polynomial in the size of the safe $\text{Eos}_k$ and Savitch's technique, which was already used in the CTL model checking procedure for safe Eos, is again applicable. In the next two subsections, we show that the results concerning CTL and also LTL model checking can be generalised to safe Eos with bounded nesting depth.

## LTL Model Checking of Safe $\text{Eos}_k$

To adapt the LTL model checking procedure from Chapter 5, markings, formulas, and the states of the constructed automata need to be of polynomial space in the size of the net again. Then, given a safe $\text{Eos}_k$ $OS$, polynomial space in the size of $OS$ must suffice to decide if a given marking $\mu$ is a deadlock and also to decide, given two markings $\mu$ and $\mu'$ and an event $\vartheta$, if $\mu \xrightarrow[OS]{\vartheta} \mu'$ holds, i.e. the Lemmata 5.20 and 5.21 have to be generalised to safe $\text{Eos}_k$.

At first, the set of atomic propositions is adapted to the new setting. Let

$$prop := \{p_0[p_1[\ldots[p_m[x]]\ldots]] \mid \begin{array}{l} m \leq k-1, x \in \{\mathbf{0}, *\}, p_0 \in \widehat{N}, \text{ and} \\ p_{i+1} \in P(d(p_i))\ \forall i : 0 \leq i < m \end{array} \} \cup \{\mathbf{0}\}.$$

As in Chapter 5 a '$*$' denotes that the place is marked arbitrarily, while $\mathbf{0}$ denotes that the place is marked with an empty token, which might also be a black token.

To estimate the size of a marking, imagine a tree. The root node has up to $|\widehat{P}|$ children, the places that are marked in the system net. Each of these nodes has as children again the places that are marked in the respective net token. With $n := \max\{|P_N| \mid N \in \widehat{\mathcal{N}}\}$ there are thus up to $n^i$ nodes at the $i$th level and with the fix nesting depth $k$ the number of nodes of the tree is bounded by $k \cdot n^k$. A marking $\mu$ could either be represented as such a tree or as up to $n^k$ elements of *prop* above each of length up to $k$, which corresponds to the paths in the tree from the root to a leaf. This is similar to the way markings were described in Chapter 5 and a computation is then a sequence of subsets of *prop* again. In any way, the size of a marking is clearly polynomial in the size of the object net system and thus also the states of the automata constructed in the LTL model checking procedure are polynomial in the size of the net system and the size of the formula (also see the discussion before Lemma 5.20).

As for safe Eos the two interesting points are thus, if it is possible to check in the required space bounds whether a final state in the automaton $A$ (resp. $B$) is reached and if it is possible to check $\mu \xrightarrow[OS]{\vartheta} \mu'$ given two markings $\mu, \mu'$ of $OS$ and an event $\vartheta$.

In case of final states, the most important point is to check if a marking $\mu$ of the Eos$_k$ $OS$, which is a state of the automaton $A_{OS}$, is a deadlock, i.e. no event is enabled in $\mu$. This is indeed possible in polynomial space again.

**Lemma 6.10.** *To check if a marking $\mu$ of a safe* Eos$_k$ *$OS$ is a deadlock, is possible in space polynomial the size of $OS$.*

*Proof.* Using nondeterminism two transitions $t$ and $t'$ are guessed and two submarkings $\lambda$ and $\lambda'$ of the current marking $\mu$. It is then checked if $t$ and $t'$ have matching labels and, if so, if $\lambda$ and $\lambda'$ fulfil the conditions of the enabling predicate. If so, $\rho$ and $\rho'$ can be constructed according to the enabling predicate. If this is possible without contradiction there is no deadlock.

Alternatively, it is possible to iterate through the transitions. For each transition $t$ another transition $t'$ with a matching label is located, if possible. Then the markings $\lambda, \lambda', \rho$, and $\rho'$ are constructed with regard to the transitions $t$ and $t'$, the appropriate enabling predicate, and with regard to the current marking $\mu$.

In both cases, the calculations simplify, if an autonomous event is enabled. Since the submarkings $\lambda$ and $\lambda'$ are smaller than $\mu$ and only some lookups

and basic mathematical operations like addition and subtraction are necessary, the whole test is possible in polynomial space in the size of the object net system. □

Finally, an analogon to Lemma 5.21 is needed.

**Lemma 6.11.** *Given a safe* $\text{EOS}_k$ *OS, two markings* $\mu$ *and* $\mu'$*, and an event* $\vartheta$*, it is possible to decide in polynomial space in* $|OS|$ *if* $\mu \xrightarrow[OS]{\vartheta} \mu'$ *holds.*

*Proof.* Since $OS$ is a safe $\text{EOS}_k$, it is possible, similar to the proof of Lemma 5.21, to calculate the submarkings $\lambda$ and $\lambda'$ from $\mu$ and the event $\vartheta$ according to the enabling predicate. The possible modes differ only in $\rho$ and $\rho'$ and there only in the distribution of the tokens to net tokens of the same type but on different places in the postsets of the firing transitions. Using $\mu'$ it is possible to calculate the correct mode, if one exists. Since only a constant number of additional temporal variables are needed, the whole procedure can be implemented in the stated space bound. □

With Lemmata 6.10 and 6.11 above LTL model checking can be accomplished for safe $\text{EOS}_k$. In particular, Algorithm 2 and Theorem 5.22 can be carried over to the setting here and thus the following theorem and corollary hold:

**Theorem 6.12.** *Given a safe* $\text{EOS}_k$ *OS and an LTL formula* $\phi$*, checking whether OS satisfies* $\phi$ *can be done in polynomial space in the size of OS and* $\phi$*, that is, there is a polynomial* $p$*, independent of OS and* $\phi$*, such that the algorithm uses* $O(p(|OS| + |\phi|))$ *space.*

**Corollary 6.13.** *The reachability problem for safe* $\text{EOS}_k$ *is* PSPACE*-complete.*

## CTL Model Checking of Safe $\text{EOS}_k$

CTL model checking of safe EOS is treated in Section 5.3. The approach there can be carried over with small adaptions to the case of safe $\text{EOS}_k$, most notably because the state space of a safe $\text{EOS}_k$ of size $n$ is bounded by $2^{p(n)}$ where $p$ is a polynomial. This polynomial is, in general, of a higher order than in the case of safe EOS, but it is still a polynomial.

At first the set of atomic propositions is extended by the set of events:

$$prop := \{p_0[p_1[\ldots[p_m[x]]\ldots]] \mid \begin{array}{l} m \leq k-1, x \in \{\mathbf{0}, *\}, p_0 \in \widehat{N}, \text{ and} \\ p_{i+1} \in P(d(p_i))\ \forall i : 0 \leq i < m \end{array} \}\cup\{\mathbf{0}\}\cup\Theta$$

The semantic of Definition 5.24 is easily adapted to the new setting. Of course, to check if $OS, \mu \models p$ holds, becomes more intricate if $p$ is a nested sequence of places $p = p_0[p_1[\ldots[p_m[\mathbf{0}]]$, but it is still possible in polynomial time

in the size of $OS$. If a marking is represented as elements of *prop*, this is clearly possible, if a marking is represented as a tree, as described in the treatment of LTL model checking above, it is only necessary to follow an appropriate path from the root, which is obtained from the proposition itself.

The interesting part is, if $OS, \mu \models p$ can still be decided in polynomial time, if $p$ is an event. By the procedure outline in the proof of Lemma 6.11 this is also possible and thus a lemma similar to Lemma 5.25 can be established:

**Lemma 6.14.** *Let $OS$ be a safe* EOS$_k$*, $\mu$ a marking of $OS$ and $p \in prop$. It is possible to decide in polynomial time in the size of $OS$ if $OS, \mu \models p$ holds.*

Since Lemma 5.26 can be carried over by use of Lemma 6.11 the main procedure outlined in Algorithm 3 in Section 5.3 can also be used for safe EOS$_k$.

Analogous statements to Lemmata 5.27 and 5.29 can also be established. The bound, however, is no longer $2^{n^2+n}$, but $2^{(n+1)^k} = 2^{O(n^k)}$ as evident in the proof of Theorem 6.9 above.

**Lemma 6.15.** *Let $OS$ be a safe* EOS$_k$*, let $n := \max\{|P_N| \mid N \in \widehat{\mathcal{N}}\}$, and let $\mu$ be a node of $RG(OS)$. Then*

$$\begin{aligned} OS, \mu \models E[\phi_1 U \phi_2] \quad &\Longleftrightarrow \quad OS, \mu \models E[\phi_1 U_{2^{(n+1)^k}} \phi_2] \\ OS, \mu \models EG\phi \quad &\Longleftrightarrow \quad OS, \mu \models EG_{2^{(n+1)^k}} \phi. \end{aligned}$$

From this it follows that the Algorithms 4, 5, and 6 can be used for safe EOS$_k$, too, where the upper bound for the number $k$ in the Algorithms 4 and 6 has to be changed to $2^{(n+1)^k}$.

These algorithms can be implemented using only polynomial space in the size of the safe EOS$_k$ and the CTL formula $\phi$ (see Lemmata 5.28 and 5.30). However, recapitulating the discussion before Lemma 5.28 the space needed is now in $O(\max\{s(\phi_1), s(\phi_2)\} + |OS|^k \cdot |OS|^k)$ for `checkEU`$(\mu, \phi_1, \phi_2)$ and, respectively, in $O(s(\phi) + |OS|^k \cdot |OS|^k)$ for `checkEG`$(\mu, \phi)$, because of the size of the state space evident in the proof of Theorem 6.9 and in Lemma 6.15 and the larger amount of space needed to store a marking of an safe EOS$_k$. Putting it together a generalisation of Theorem 5.31 is possible:

**Theorem 6.16.** *Given a safe* EOS$_k$ *$OS$ and a CTL formula $\phi$ checking whether $OS$ satisfies $\phi$ can be done in $O(|OS|^{2k} \cdot |\phi|)$ space.*

Since $k$ is a constant the bound given in the theorem above is polynomial. Moreover, since Algorithm 7 can also be adapted, liveness can be decided in polynomial space for safe EOS$_k$:

**Corollary 6.17.** *The liveness problem for safe* EOS$_k$ *is* PSPACE*-complete.*

In summary, LTL and CTL model checking of safe EOS$_k$ is possible in a similar way as for safe EOS. The polynomial depends on $k$ and worsens with increasing $k$, but since $k$ is a constant, the procedures only need polynomial space in the size of the EOS$_k$ and the formula.

## 6.3 Safeness for ONS

Turning again to ONS, a safeness definition similar to the one for $\text{EOS}_k$, which lead to algorithms in PSPACE, is desirable. The condition from Definition 6.8

$$\forall \mu \in R(OS, \mu_0) : \forall \mu' \in \triangledown(\mu)^* : \forall p \in \mathcal{P} : \Pi^1(\mu')(p) \leq 1$$

ensures for ONS, as well, that in every reachable marking there is at most one token on each place. However, due to the unrestricted nesting depth the state space might become infinite nonetheless. Figure 6.9 shows an example. The system net consists of only one place $\widehat{p}$ and there is only one object net type $N$. The initial marking is $\widehat{p}[p_1[\mathbf{0}]]$. The net token on place $\widehat{p}$ may now synchronise with the net token on its place $p_1$, i.e. the event $(t_2, t_1)$ is enabled and may fire resulting in the successor marking $\widehat{p}[p_2[p_1[\mathbf{0}]]]$, i.e. in the net token on the deepest nesting depth an empty net token was created on place $p_1$. The situation is now similar as before and the event $(t_2, t_1)$ is again activated, this time at a different nesting level. Firing again results in the marking $\widehat{p}[p_2[p_2[p_1[\mathbf{0}]]]]$. This can be repeated infinitely often, resulting in an infinite state space. However, each net token is 1-safe, if the inner marking of the nested tokens are ignored, i.e. the condition above holds.

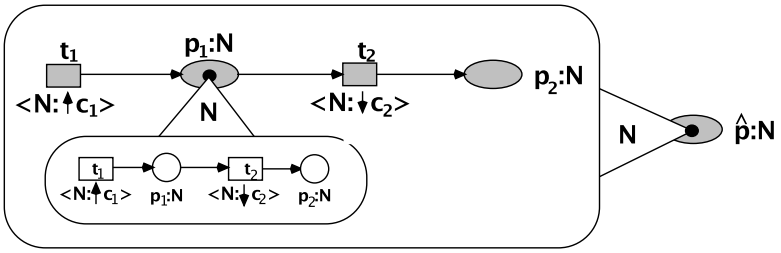


Figure 6.9: An ONS whose net tokens are all 1-safe but whose reachability set is infinite nonetheless.

For safe ONS finiteness of the state space is therefore explicitly required:

**Definition 6.18** (**Bounded and Safe ONS**). *Let $OS$ be an ONS with initial marking $\mu_0$. $OS$ is* bounded *iff $|R(OS, \mu_0)| < \infty$, i.e. if the set of reachable markings of $OS$ is finite.*

*$OS$ is* safe *iff $OS$ is bounded and there is at most one token on each place, that is, if $|R(OS, \mu_0)| < \infty$ and*

$$\forall \mu \in R(OS, \mu_0) : \forall \mu' \in \triangledown(\mu)^* : \forall p \in \mathcal{P} : \Pi^1(\mu')(p) \leq 1$$

While the state space of a safe ONS is now finite by definition and every net token is 1-safe with regard to the projection $\Pi^1$, the state space might, unfortunately, still become very big.

Figure 6.10 shows an example. The system net is given by $\widehat{N} = N_0$ on the upper left. Then there are $k-1$ object nets $N_1, \ldots, N_{k-1}$ with identical structure, but differing labeling. They are given at the lower left. At last, there is an object net $N_k$ given to the right of Figure 6.10. The initial marking is $\widehat{p}[\mathbf{0}]$, i.e. the system net place $\widehat{p}$ is marked by a black token. By firing the transition to the left, two empty net tokens of type $N_1$ are created on the places $\widehat{p}_1$ and $\widehat{p}_2$. By firing the next two transitions in the system net, in the net tokens of type $N_1$ two net tokens of type $N_2$ are created and the tokens are moved to $\widehat{p}'_1$ and $\widehat{p}'_2$, respectively. This is continued and finally there are $2^k$ net tokens of type $N_k$ each marked with two black tokens. Each of these black tokens can now be removed or not resulting in $2^{2^{k+1}}$ possibilities. Thus, the size of the state space is bounded from *below* by $2^{2^{k+1}}$ and since $k$ is the number of net token types and the size of the net system is polynomial in $k$, the state space is exponential in the size of the net system.

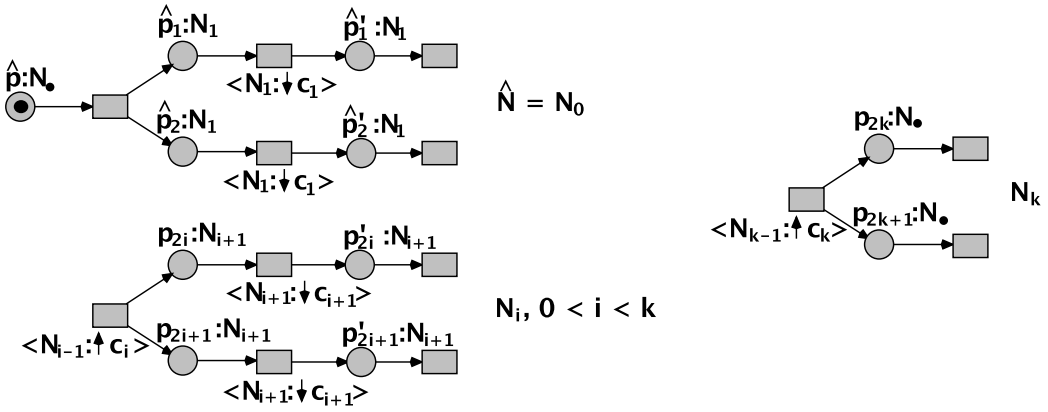


Figure 6.10: A safe ONS with a state space of exponential size.

Savitch's technique to solve the reachability problem is still applicable in this case, but since $\log 2^{2^{k+1}} = O(2^k)$, the calculation is not possible in polynomial space anymore. Indeed, it seems that the reachability problem for safe ONS is far beyond PSPACE and requires exponential time or worse. We conjecture:

**Conjecture 6.19.** *The Reachability problem for safe ONS is* EXPTIME*-complete.*

Additionally forbidding the creation of net tokens, reduces the size of the state space considerably and, even if the state space is often larger than in the case of $\text{EOS}_k$ due to the different possible nestings, allows to solve the reachability problem in PSPACE again. This restriction is similar to the restriction employed for GSM (see Definition 4.6) and is indeed not as severe as it might seem at first glance. The following definition 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 6.20.** *Let OS be an ONS and* $\mu = \sum_{k=1}^{n} p_k[M_k]$ *be a marking of OS. With* $\Pi^3_N(\mu)$ *we denote the* number *of net-tokens of type N present in* $\mu$*, i.e.*

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

$\Pi^3_N(\mu)$ *is thus calculated recursively, where* $\Pi^3_N(\mathbf{0}) = \mathbf{0}$.

**Definition 6.21** (**Strongly safe ONS**). *Let OS be an ONS with initial marking* $\mu_0$*. OS is* strongly safe *iff OS is safe and* $\Pi^3_N(\mu) = \Pi^3_N(\mu')$ *holds for all* $\mu, \mu' \in R(OS, \mu_0)$ *and for all* $N \in \mathcal{N} \setminus \{N_\bullet\}$*, i.e. no net tokens are created nor destroyed*

**Theorem 6.22.** *Let OS be a strongly safe ONS. The size of the state space of OS is bounded by* $2^{p(|OS|)}$ *where p is a polynomial.*

*Proof.* 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 $k$ 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 $n^{n-2}$ [Cay89]. 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). This bound is only a rough approximation, but it suffices here.

Taking the number $m := |\widehat{P}|$ of system net places into account we end up with at most $(k^{2k} \cdot 2^{nk})^m \leq (k^{2k} \cdot 2^{nk})^n = k^{2kn} \cdot 2^{nkn} = 2^{\log k \cdot 2kn} \cdot 2^{nkn} < 2^{2kkn+nkn}$ markings. Since $2kkn + nkn$ is clearly a polynomial in the input length, the proof is complete. □

With the established space bound, the approach used to show that LTL and CTL model checking is in PSPACE for safe $\text{EOS}_k$ can be adapted. A strongly safe ONS with $k$ net tokens can in many cases be seen as an $\text{EOS}_k$, because no tokens are created in the ONS and thus the $k$ net tokens might at most be nested into each other to create a nesting of depth $k$. As set of atomic propositions it is thus possible to take the same sets used in the case of safe $\text{EOS}_k$. Lemmata 6.10 and 6.11 and subsequently also Theorem 6.12 and Corollary 6.13 then carry over to strongly safe ONS easily. In the proofs it is only necessary to take the two firing predicates into account, too, which where not important in the case of safe $\text{EOS}_k$ (see equations 6.1 and 6.2). For these only a few more tests are necessary that can easily be done in polynomial space. Hardness follows from the proof of Lemma 4.20. There it is shown that the reachability problem is PSPACE-hard for ppGSMs and this proof can be carried over one-to-one to strongly safe ONS.

For CTL model checking the Lemmata 6.14 and 6.15 can also be carried over, albeit in Lemma 6.15 the bounds change according to the bound established in the proof of Theorem 6.22 above. The exponent of this bound, however, is still a polynomial in the input size and thus Theorem 6.16 and Corollary 6.17 can be carried over to strongly safe ONS, as well.

Thus the following theorem and corollary hold for strongly safe ONS:

**Theorem 6.23.** *Given a strongly safe ONS OS, a CTL formula $\phi$, and a LTL formula $\psi$. Checking whether OS satisfies $\phi$ can be done in polynomial space in the size of OS and $\phi$. Checking whether OS satisfies $\psi$ can be done in polynomial space in the size of OS and $\psi$.*

**Corollary 6.24.** *For strongly safe ONS the reachability and the liveness problem are* PSPACE*-complete.*

Comparing strongly safe ONS and safe EOS as modelling languages, strongly safe ONS are on the one hand enhanced, since the vertical transport of tokens is possible. On the other hand, they are restricted, since the creation and destruction of net tokens is forbidden. Strongly safe ONS thus have additional modelling features, on the one hand, and lack some other modelling features, on the other. But due to the restrictions, LTL and CTL model checking procedures can again be implemented, using only polynomial space. Strongly safe ONS can thus be seen as an alternative to safe EOS in certain cases where the additional modelling capability is of interest.

## 6.4 Summary

Due to the results from Chapter 4 and 5 safe EOS are established as a formalism with, in comparison to EOS, no structural restrictions and a safeness restrictions

guaranteeing that the size of the state space is finite and that LTL and CTL model checking are possible in polynomial space.

Diverging from the search for sensible restrictions of Eos in the last two chapters, we enhanced Eos in this chapter.

We introduced ONS which give an modeller the possibility to model certain applications where the vertical transport of tokens is of importance. For example, if an object that is already modelled by a net token is put into another object also modelled by a net token. As restrictions of ONS, extensions of Eos and GSMs were introduced, namely $\text{Eos}_k$, $\text{GSM}_k$, i.e. Eos and GSMs with an arbitrary but fixed nesting depth.

ONS and $\text{Eos}_k$ are unsurprisingly Turing-complete due to the results for Eos. However, introducing a safeness concept similar to the one for safe Eos it is again possible to decide in polynomial space, given a safe $\text{Eos}_k$ *OS* and a CTL or LTL formula $\phi$, whether *OS* satisfies $\phi$ or not.

For ONS a similar safeness notion as for $\text{Eos}_k$ leads to a state space size that is not bounded in such a way that the techniques used before are again applicable. However, a stronger safeness notion was introduced and for these strongly safe ONS it is again possible to decide the LTL and CTL model checking problem in polynomial space.

With these three formalisms, namely safe Eos, safe $\text{Eos}_k$, and strongly safe ONS, three formalisms have been established which allow to model applications in which mobility, nesting, and interaction of objects are important, but which also allow to verify properties of the model expressed in LTL or CTL in polynomial space. Safe Eos have a smaller polynomial bound in comparison with safe $\text{Eos}_k$, which in exchange allow a deeper nesting of net tokens. In strongly safe ONS the nesting might then change and the net tokens may also travel in the vertical dimension, but no creation of net tokens is allowed.

Finally, we want to point out that the temporal logics LTL and CTL have their drawbacks when it comes to ONS. Although LTL and CTL are quite expressive, it is only possible to reason about time with them. In the setting here, the locations of object nets become important as well. One might want to specify that a certain agent is at least *somewhere* with regard to the places and the nesting. However, to describe this with CTL or LTL one might end up designing an formula of infinite length.

A logic is needed that allows to reason about the nesting and the location of net tokens of an object net system. Inspired by the work of Cardelli and Gordon on their Ambient Logic, introduced for the Ambient Calculus, a process calculi that allows a nesting of process terms (cf. [CG00b] and [CG00a]), an operator "somewhere" could be introduced to describe that a formula holds at some nesting level of the current marking. Let $\boxdot$ be this new operator. The semantic could be defined as follows:

$$\forall \mu \in \mathcal{M} \quad \mu \models \boxdot\phi \quad \text{iff} \quad \exists \mu' : \mu' \in \triangledown(\mu)^* \wedge \mu' \models \phi$$

The relation $\mu \models \circledcirc p[*]$ holds if in the marking $\mu$ an object net $N$ resides (in some nesting depth) whose place $p$ is marked (in an arbitrary way). With a "finally" operator $F$ as for CTL or LTL the formula $F \circledcirc p[T]$ is satisfied in a marking $\mu$, if from $\mu$ a marking $\mu'$ is reachable that satisfies $\circledcirc p[T]$.

Such a logic allows us then not only to reason about the evolution of the described system in time, but also about spatial configurations, about locations, and, in particular, about the nesting of object nets.

The formalism of ONS has been published as joint work with Michael Köhler-Bußmeier in [KBH09] in a different form. The firing rule there was more complicated and whole trees of transitions where allowed to fire. The formalism of ONS as presented here is published in [HKB12a]. It restricts the transitions participating in an event to only two adjacent levels.

The notions and results from Sections 6.2 and 6.3 are, apart from a preliminary result in [HKB12a], presented here for the first time.

A first sketch of a logic with the properties discussed above can also be found in [HKB12a]. This work is inspired by the work of Cardelli and Gordon on the Ambient Calculus and the Ambient Logic [CG00b], [CG00a].

# 7 Conclusion

A number of formalisms are known by now that in one way or the other apply the concept of nesting to Petri nets (or process algebraic terms). These formalisms aim at capturing the idea of nested structures and mobility of objects and, by design of the firing rule, the idea of interaction between these objects.

Mobility, interaction, and the nesting of structures are probably some of the most important concepts when designing future computational systems. Due to the various possible interactions and movements, these systems tend to be complicated. Hence, suitable modelling languages are of great importance, as is the possibility to automatically verify properties of the model.

These modelling languages include elementary object systems, which have been in the focus of this thesis. Being Turing-complete in its general form, EOS are suitable to model applications in, e.g., an agent context, but verifying properties of the model automatically is out of reach. This is a severe drawback to the usability of this formalism in practice.

Consequently, the main goal of this thesis is to restrict the formalism in such a way that modelling is still comfortably possible, but important problems become decidable – and that with as little usage of resources as possible. The focus of our attention was the reachability problem as one of the most important and also most basic verification problems.

This goal has successfully been reached in terms of the following results: Structurally restricting the participating nets alone, is not enough if one strives for a formalism in which the reachability problem is easily solvable. This is evident in, e.g., the ppGSMs and the free-choice GSMs investigated. In both formalisms the restriction to the system net and to the object nets are such that, if treated as p/t nets, the reachability problem is solvable in polynomial time, yet the ability to interact worsens the complexity to PSPACE-hardness in the case of ppGSMs and even to EXPSPACE-hardness in the case of free-choice GSMs. Even after restricting the possibilities to interact by utilising the restrictions devised for the definitions of deterministic and strongly deterministic GSMs, these complexity bounds remain.

The restriction to safe EOS that restricts the set of reachable markings and is thus a dynamic restriction, then leads to the surprising result that not only reachability but every property that can be expressed in the temporal logics LTL or CTL can be verified in polynomial space. This result holds without any further structural restriction.

Table 7.1: Complexity of the reachability problem for various structurally restricted formalisms.

| | strongly deterministic | deterministic | general |
|---|---|---|---|
| ttGSM | P | ? | ? |
| ppGSM | PSPACE-complete | PSPACE-complete | PSPACE-complete |
| ptGSM | NP-hard | PSPACE-hard | PSPACE-hard |
| tpGSM | ? | ? | ? |
| acGSM | NP-hard | NP-hard | NP-hard |
| cfGSM | NP-complete | NP-hard | NP-hard |
| fcGSM | EXPSPACE-hard | EXPSPACE-hard | EXPSPACE-hard |
| GSM | EXPSPACE-hard | EXPSPACE-hard | EXPSPACE-hard |
| cEOS | undecidable | undecidable | undecidable |
| EOS | undecidable | undecidable | undecidable |

Table 7.2: Complexity of the reachability problem for *system safe* EOS with further structural restrictions.

| | strongly deterministic | deterministic | general |
|---|---|---|---|
| ttGSM | P | ? | ? |
| ppGSM | PSPACE-complete | PSPACE-complete | PSPACE-complete |
| ptGSM | NP-hard | PSPACE-hard | PSPACE-hard |
| tpGSM | ? | ? | ? |
| acGSM | NP-hard | NP-hard | NP-hard |
| cfGSM | NP-complete | NP-hard | NP-hard |
| fcGSM | EXPSPACE-hard | EXPSPACE-hard | EXPSPACE-hard |
| GSM | EXPSPACE-hard | EXPSPACE-hard | EXPSPACE-hard |
| cEOS | undecidable | undecidable | undecidable |
| EOS | undecidable | undecidable | undecidable |

Also evident in the investigations is, that additional structural restrictions, i.e. considering EOS that are safe *and also* structurally restricted, are not only unnecessary, but actually do have little to none effect in most cases. The threshold only seldom drops below PSPACE. Moreover, weaker safeness restrictions, like in the definition of system safeness, where the restriction mainly lies with the system net, are not enough. The thresholds are in the cases investigated identical to the general formalisms without the requirement of system safeness.

The results concerning the complexity of the reachability problem for the various introduced and studied formalisms are summarised in Tables 7.1, 7.2, and 7.3. The abbreviation cEOS is used for conservative EOS. Note that unless stated otherwise, the reachability problem is decidable for the considered formalism.

The severe lower bounds for the structurally restricted formalisms are evident in Table 7.1. Table 7.2 lists the results for system safe formalisms. The results

Table 7.3: Complexity of the reachability problem for *safe* Eos with further structural restrictions.

| | strongly deterministic | deterministic | general |
|---|---|---|---|
| ttGSM | P | PSpace | PSpace |
| ppGSM | PSpace-complete | PSpace-complete | PSpace-complete |
| ptGSM | NP-hard, PSpace | PSpace-complete | PSpace-complete |
| tpGSM | PSpace | PSpace | PSpace |
| acGSM | PSpace | PSpace | PSpace |
| cfGSM | P | PSpace | PSpace |
| fcGSM | PSpace-complete | PSpace-complete | PSpace-complete |
| GSM | PSpace-complete | PSpace-complete | PSpace-complete |
| cEos | PSpace-complete | PSpace-complete | PSpace-complete |
| Eos | PSpace-complete | PSpace-complete | PSpace-complete |

Table 7.4: Complexity of the reachability problem for further formalisms

| | unary Eos | persistent Eos | semi-bounded Eos |
|---|---|---|---|
| reachability | undecidable | undecidable | ExpSpace-hard |

obtained are identical, but it might be possible that the bounds diverge a little in further investigations. Note that while reachability and also liveness was shown to be undecidable for conservative and also system safe and conservative Eos, boundedness becomes decidable for these net classes. Nonetheless, with the bounds obtained the formalisms in Tables 7.1 and 7.2 are hardly useable for modelling, if there is a need to verify properties of the created models.

The results concerning the safe formalisms are summarised in Table 7.3. A main result of this thesis is, that in case of a PSpace-bound, not only the reachability problem is solvable in PSpace, but every property expressible in LTL or CTL.

Table 7.4 lists further investigated formalisms not central to the study here.

The conclusion of this work is thus that, if possible, a safe formalism should be used to model a system. This model can then be analysed with algorithms that run in acceptable time and space bounds. Moreover, no further structural restrictions are necessary in case of a safe formalism.

The relevance of such verification algorithms can hardly be overestimated. The models of systems, that employ mobility and interaction of diverse entities, are usually complex. Thus not all details and implications can be seen easily, not even by the designer him- or herself. It is thus of great significance to be able to verify automatically if certain properties are satisfied.

Furthermore, although it is quite often possible that a modeller can guarantee that his or her model is safe, another useful advantage of the safe Eos is, that it is also possible to verify automatically in PSpace if a given model is safe or not.

Table 7.5: Complexity of the reachability problem for ONS-like formalisms

| | ONS | strongly safe ONS | safe $\text{EOS}_k$ | safe $\text{GSM}_k$ |
|---|---|---|---|---|
| reachability | undecidable | PSPACE-complete | | |

Thus the goal stated in the introduction to restrict the formalism of object nets in such a way that important problems become decidable, has been reached. To ascertain that the formalisms proposed are also useful for the modelling of systems, models of practical relevant applications need to be created with them and the benefit of the models need to be evaluated. While it is likely that the examples addressed in the introduction can be modelled with safe EOS, as well, this would have to be done in detail. This, however, as well as the creation of models of other applications is out of the scope of this dissertation and is a topic in its own right.

Diverging from the main branch of this work, in Chapter 6 object net systems are introduced as an *extension* of elementary object systems and not a restriction. They allow the vertical transport of tokens and hence an arbitrarily nesting depth. EOS resp. GSMs with an arbitrary but fixed nesting depth are furthermore introduced as an restriction of object net systems that do not allow the vertical transport of tokens.

From a modelling point of view, these formalisms are quite attractive, since they allow a true exchange of net tokens and thus take the concept of mobility one step further.

Since they are an extension of EOS, structural restrictions will be of little help: The negative results from structurally restricted EOS will carry over to the new formalism. However, safeness concepts similar to the concepts for EOS and GSMs can be established and it turns out that, albeit the polynomial worsens, reachability can again be decided using only polynomial space. Moreover, similar results for LTL and CTL model checking can again be established, i.e. not only reachability, but also every other property expressible in LTL or CTL can be checked for safe $\text{EOS}_k$, safe $\text{GSM}_k$, and strongly safe ONS using only polynomial space in the size of the net system and the formula. These formalisms can thus be seen as an alternative to safe EOS also allowing verification procedures in PSPACE, but offering different modelling capabilities. The results for ONS and restricted variants are summarised in Table 7.5.

Altogether, almost 100 different formalisms are treated in this thesis. Some of these, namely EOS, conservative EOS, and GSMs have been known before the work presented in this thesis, but most of them are new, as are the important definitions of determinism and safeness, the thorough analysis of the reachability problem for these formalisms, and the main results about LTL and CTL model checking of safe $\text{EOS}_k$, safe $\text{GSM}_k$, and strongly safe ONS.

We have arrived at formalisms that we believe will prove helpful, allowing the modelling of many applications involving systems in systems, moving objects,

and communication, while still permitting algorithmic verification and analysis of the model used employing only affordable resources.

## Outlook and Open Questions

The unsolved cases in the tables above, i.e. the cases where the upper and lower bounds do not match or one or both of them are unknown, are naturally open questions. However, from a practical point of view and with regard to the subject of this thesis, solving these questions will only have little impact. The lower bounds in Table 7.1 are in most cases already so high that finding an algorithm and thus an upper bound will hardly be of practical relevance. Pinpointing the exact complexity of the safe formalisms (see Table 7.3) is of more practical value, since in some cases the threshold might drop below PSPACE. However, the structural restrictions are in most cases quite severe and it is thus doubtful that such a restricted formalism is useful for modelling. It seems that in case of a safe EOS with only modest structural restrictions the reachability problem is promptly PSPACE-hard and thus the PSPACE-algorithm proposed is optimal with regard to complexity classes.

Nevertheless, research into algorithms for the reachability problem or other verification problems for these object net formalisms might shed new light onto the formalisms and help to find other suitable restrictions for which faster algorithms can be designed. Furthermore, the PSPACE-algorithms described in this thesis can surely be improved. The resulting algorithms will still run in PSPACE, but the polynomial will be smaller.

For the design of better algorithms it might also be interesting to investigate the differences between safe and system safe and especially between safe(3) and safe(4) in more detail. In the first case, the practical relevance is again limited, because the lower bounds known are already quite bad. In the second case one will not be able to bypass the PSPACE-bound, but the bounds in the case of safe(4) EOS might nonetheless be better than in the case of the more liberal safe(3) EOS.

Another line of research only partly covered by the restrictions investigated here is concerned with *compositionality*. Here the goal is to define restrictions such that problems for the whole system can be solved by examining the system's components in isolation. Dworzański and Lomazova recently investigated what requirements are necessary in their nested nets formalisms such that boundedness and liveness can be decided in a compositional way [DL11, DL12]. The requirements are quite involved and seem too strong, but they give a first important result in this area.

What requirements could be imposed upon EOS and object net systems to solve problems in a similar compositional way, is an open question, but it is likely that the results in this matter would be beneficial for both the modeller and the verifier. The modeller would probably benefit from a more

rule-based modelling framework and the verifier would be able to design efficient algorithms.

Other approaches, successfully used in the verification of systems, like partial order model checking or symbolic model checking, might also be transferred or applied to object net systems, but each such topic is probably a thesis in its own right.

Finally, from a practical point of view, it would be nice to have tool support for the modelling of systems with the formalisms presented here. In these tools the verification algorithms designed in this thesis and future algorithms could be implemented so that the models could be verified automatically. With such a tool the design and construction of mobile systems could be dramatically simplified. With the results for safe Eos and strongly safe ONS, this thesis has laid the fundamentals for the algorithmic aspects of such a tool.

# Bibliography


[BBPP05] Marek A. Bednarczyk, Luca Bernardinello, Wiesław Pawłowski, and Lucia Pomello. Modelling mobility with Petri hypernets. In Josè Luiz Fiadeiro, Peter D. Mosses, and Fernando Orejas, editors, *Recent Trends in Algebraic Development Techniques. 17th International Workshop, WADT 2004, Barcelona, Spain, March 27-29, 2004. Revised Selected Papers*, volume 3423 of *Lecture Notes in Computer Science*, pages 28–44. Springer-Verlag, 2005.

[BJP06] Marek A. Bednarczyk, Wojciech Jamroga, and Wiesław Pawłowski. Expressing and verifying temporal and structural properties of mobile agents. *Fundamenta Informaticae*, 72(1-3):51–63, 2006.

[BJP12] Marek A. Bednarczyk, Piotr Jóźwiak, and Wiesław Pawłowski. A class of hypernets with token creation and decidable reachability problem. In Louchka Popova-Zeugmann, editor, *Proceedings of the 21st International Workshop on Concurrency, Specification, and Programming (CS&P 2012)*, volume 928, pages 37–48. CEUR Workshop Proceedings, 2012.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, Cambridge, Massachusetts, 2008.

[BT87] Eike Best and P.S. Thiagarajan. Some classes of live and save Petri nets. In K. Voss, H. J. Genrich, and G. Rozenberg, editors, *Concurrency and Nets*, Advances in Petri Nets, pages 71–94. Springer-Verlag, 1987.

[Cay89] Arthur Cayley. A theorem on trees. *Quarterly Journal of Pure and Applied Mathematics*, 23:376–378, 1889.

[CCGR99] A. Cimatti, E.M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: a new symbolic model verifier. In N. Halbwachs and D. Peled, editors, *Proceedings Eleventh Conference on Computer-Aided Verification (CAV'99)*, volume 1633 of *Lecture Notes in Computer Science*, pages 495–499. Springer-Verlag, 1999.

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Proceedings Logics of Programs, Workshop, Yorktown Heights, New York, May 1981*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[CEP93] Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity results for 1-safe nets. In *Proceedings of the 13th International Conference on the Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*, pages 326–337. Springer-Verlag, 1993.

[CEP95] Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147:117–136, 1995.

[CG00a] Luca Cardelli and Andrew D. Gordon. Anytime, anywhere. modal logics for mobile ambients. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 365–377. ACM Press, 2000.

[CG00b] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240:177–213, 2000.

[CH94] Søren Christensen and Niels Damgaard Hansen. Coloured Petri nets extended with channels for synchronous communication. In Robert Valette, editor, *Application and Theory of Petri Nets 1994, 15th International Conference, Zaragoza, Spain, June 20-24, 1994, Proceedings*, volume 815 of *Lecture Notes in Computer Science*, pages 159–178. Springer-Verlag, 1994.

[CHEP71] F. Commoner, A. W. Holt, S. Even, and A. Pnueli. Marked directed graphs. *Journal of Computer and System Sciences*, 5:511–523, 1971.

[CLM76] E. Cardoza, Richard J. Lipton, and Albert R. Meyer. Exponential space complete problems for Petri nets and commutative semigroups. In Ashok K. Chandra, Detlef Wotschke, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the 8th Annual ACM Symposium on the Theory of Computing (STOC'76)*, pages 50–54. ACM, 1976.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, third edition, 2009.

[CT12] Frédéric Cristini and Catherine Tessier. Nets-within-nets to model innovative space system architectures. In Serge Haddad and Lucia Pomello, editors, *Application and Theory of Petri Nets. 33rd International Conference, PETRI NETS 2012. Hamburg, Germany, June 2012. Proceedings*, volume 7347 of *Lecture Notes in Computer Science*, pages 348–367. Springer-Verlag, 2012.

[CVN05] G. Castagna, J. Vitek, and F. Zappa Nardelli. The seal calculus. *Information and Computation*, 201:1–54, 2005.

[CZG+03] Witold Charatonik, Silvano Dal Zilio, Andrew D. Gordon, Supratik Mukhopadhyay, and Jean-Marc Talbot. Model checking mobile ambients. *Theoretical Computer Science*, 308:277–331, 2003.

[DE91] Jörg Desel and Javier Esparza. Reachability in reversible free choice systems. In Christian Choffrut and Matthias Jantzen, editors, *STACS 91. 8th Annual Symposium on Theoretical Aspects of Computer Science Hamburg, Germany, February 14-16, 1991 Proceedings*, volume 480 of *Lecture Notes in Computer Science*, pages 384–397. Springer-Verlag, 1991.

[DE93] Jörg Desel and Javier Esparza. Reachability in cyclic extended free choice systems. *Theoretical Computer Science*, 114:93–118, 1993.

[DE95] Jörg Desel and Javier Esparza. *Free choice Petri nets.* Cambridge University Press, New York, NY, USA, 1995.

[DK06] Roxana Dietze and Manfred Kudlek. Subclasses of minmal based object nets with decidable reachability problem. In *Bericht 267, Tagungsband des 13. Workshops Algorithmen und Werkzeuge für Petri-Netze, AWPN'06, FBI-HH-B-267/06*, pages 32–36, 2006.

[DL11] Leonid W. Dworzański and Irina A. Lomazova. On compositionality of boundedness and liveness for nested Petri nets. In Marcin Szczuka, Ludwik Czaja, Andrzej Skowron, and Magdalena Kacprzak, editors, *Concurrency, Specification and Programming (CS&P 2011), Proceedings*, Pułtusk, Poland, 2011. Białystok University of Technology.

[DL12] Leonid W. Dworzański and Irina A. Lomazova. On compositionality of boundedness and liveness for nested Petri nets. *Fundamenta Informaticae*, 120(3–4):275–293, 2012.

[DTMZ08] Vineela Devarashetty, Jeffrey J. P. Tsai, Lu Ma, and Du Zhang. Modeling a secure sensor network system using an extended elementary object system. In Yingxu Wang, Du Zhang, Jean-Claude

Latombe, and Witold Kinsner, editors, *Proceedings of the Seventh IEEE International Conference on Cognitive Informatics, ICCI 2008, Stanford University, California, USA, August 14-16, 2008*, pages 67–74. IEEE, 2008.

[DTMZ10] Vineela Devarashetty, Jeffrey J. P. Tsai, Lu Ma, and Du Zhang. Modeling a secure sensor network using an extended elementary object system. *IJCINI*, 4(3):1–17, 2010.

[EN94] Javier Esparza and Mogens Nielsen. Decidability issues for Petri nets - a survey. *Journal of Information Processing and Cybernetics*, 30(3):143–160, 1994.

[ES92] Javier Esparza and Manuel Silva. A polynomial-time algorithm to decide liveness of bounded free choice nets. *Theoretical Computer Science*, 102:185–205, 1992.

[Esp98a] Javier Esparza. Decidability and complexity of Petri net problems – an introduction. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer-Verlag, 1998.

[Esp98b] Javier Esparza. Reachability in live and safe free-choice Petri nets is NP-complete. *Theoretical Computer Science*, 198(1):211–224, 1998.

[FEA03a] David de Frutos-Escrig and Olga Marroquín Alonso. Ambient Petri nets. *Electronic Notes in Theoretical Computer Science*, 85(1):39, 2003.

[FEA03b] David de Frutos-Escrig and Olga Marroquın Alonso. Replicated ambient Petri nets. In Peter M.A. Sloot, David Abramson, Alexander V. Bogdanov, Yuriy E. Gorbachev, Jack J. Dongarra, and Albert Y. Zomaya, editors, *Computational Science - ICCS 2003. International Conference Melbourne, Australia and St. Petersburg, Russia June 2-4, 2003 Proceedings, Part II*, volume 2658 of *Lecture Notes in Computer Science*, pages 774–783. Springer-Verlag, 2003.

[FS01] A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere. *Theoretical Computer Science*, 256(1-2):63–92, 2001.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, USA, 1979.

[GL73] H. J. Genrich and K. Lautenbach. Synchronisationsgrpahen. *Acta Informatica*, 2:143–161, 1973.

[GV03] Claude Girault and Rüdiger Valk. *Petri nets for systems engineering - a guide to modeling, verification, and applications*. Springer-Verlag, Berlin, 2003.

[GV08] Orna Grumberg and Helmut Veith, editors. *25 Years of Model Checking - History, Achievements, Perspectives*, volume 5000 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008.

[Hac72] M. Hack. Analysis of production schemata by Petri nets. Tr-94, MIT-MAC, 1972.

[Hac74] M. Hack. The recursive equivalence of the reachability problem and the liveness problem for Petri nets and vector addition systems. In *Proceedings of the 15th Annual Symposium on Switching and Automata Theory*, pages 156–164. IEEE Computer Society, 1974.

[Hac76a] M. Hack. *Decidability Questions for Petri Nets*. PhD thesis, M.I.T, 1976.

[Hac76b] M. Hack. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2:77–95, 1976.

[Hac76c] M. Hack. Petri net language. Report of the department of informatics, Cambridge, MA, USA, 1976.

[HEM05] Kathrin Hoffmann, Hartmut Ehrig, and Till Mossakowski. High-level nets with nets and rules as tokens. In Gianfranco Ciardo and Philippe Darondeau, editors, *Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005, Proceedings*, volume 3536 of *Lecture Notes in Computer Science*, pages 268–288. Springer-Verlag, 2005.

[Hir02] Kunihiko Hiraishi. $PN^2$: An elementary model for design and analysis of multi-agent systems. In Farhad Arbab and Carolyn L. Talcott, editors, *Coordination Models and Languages, COORDINATION 2002*, volume 2315 of *Lecture Notes in Computer Science*, pages 220–235. Springer-Verlag, 2002.

[HKB11a] Frank Heitmann and Michael Köhler-Bußmeier. On defining conflict-freedom for object nets. In B. Farwer and M. Köhler-Bußmeier, editors, *Proceedings of the Second International Workshop on Logic, Agents, and Mobility (LAM 2011)*, 2011.

[HKB11b] Frank Heitmann and Michael Köhler-Bußmeier. Restricting generalised state machines. In Marcin Szczuka, Ludwik Czaja, Andrzej Skowron, and Magdalena Kacprzak, editors, *Concurrency, Specification and Programming (CS&P 2011), Proceedings*, Pułtusk, Poland, 2011. Białystok University of Technology.

[HKB12a] Frank Heitmann and Michael Köhler-Bußmeier. A mobility logic for object net systems. In B. Farwer and M. Köhler-Bußmeier, editors, *Proceedings of the Third International Workshop on Logic, Agents, and Mobility (LAM 2012)*, 2012.

[HKB12b] Frank Heitmann and Michael Köhler-Bußmeier. P- and t-systems in the nets-within-nets-formalism. In Serge Haddad and Lucia Pomello, editors, *Application and Theory of Petri Nets. 33rd International Conference, PETRI NETS 2012. Hamburg, Germany, June 2012. Proceedings*, volume 7347 of *Lecture Notes in Computer Science*, pages 368–387. Springer-Verlag, 2012.

[HM03] Kathrin Hoffmann and Till Mossakowski. Algebraic higher-order nets: Graphs and Petri nets as tokens. In M. Wirsing, D. Pattinson, and R. Henicker, editors, *Proceedings of 16th Int. Workshop of Algebraic Development Techniques*, volume 2755 of *Lecture Notes in Computer Science*, pages 253–267. Springer-Verlag, 2003.

[HMU01] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Boston, MA, USA, second edition, 2001.

[HO02] Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion.* Springer-Verlag, Berlin, Deutschland, 2002.

[Hol04] Gerard J. Holzmann. *The SPIN Model Checker. Primer and Reference Manual.* Addison-Wesley, 2004.

[HP99] Serge Haddad and Denis Poitrenaud. Theoretical aspects of recursive Petri nets. In S. Donatelli and J. Kleijn, editors, *Application and Theory of Petri Nets*, volume 1639 of *Lecture Notes in Computer Science*, pages 228–247. Springer-Verlag, 1999.

[HR88] Rodney R. Howell and Louis E. Rosier. Completeness results for conflict-free vector replacement systems. *Journal of Computer and System Sciences*, 37:349–366, 1988.

[HR89] Rodney R. Howell and Louis E. Rosier. Problems concerning fairness and temporal logic for conflict-free Petri nets. *Theoretical Computer Science*, 64:305–329, 1989.

[HR04] Michael Huth and Mark Ryan. *Logic in Computer Science. Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, UK, 2nd edition, 2004.

[HRY87] Rodney R. Howell, Louis E. Rosier, and Hsu-Chen Yen. An $O(n^{1.5})$ algorithm to decide boundedness for conflict-free vector replacement systems. *Information Processing Letters*, 25:27–33, 1987.

[HT10] Christian Haubelt and Jürgen Teich. *Digitale Hardware/Software-Systeme. Spezifikation und Verifikation*. eXamen.press. Springer-Verlag, Berlin, 2010.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, USA, 1979.

[Jen97a] Kurt Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts*. Monographs in Theoretical Computer Science. Springer-Verlag, 1997. 2nd corrected printing.

[Jen97b] Kurt Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2, Analysis Methods*. Monographs in Theoretical Computer Science. Springer-Verlag, 1997. 2nd corrected printing.

[Jen97c] Kurt Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 3, Practical Use*. Monographs in Theoretical Computer Science. Springer-Verlag, 1997.

[JK09] Kurt Jensen and Lars M. Kristensen. *Coloured Petri Nets. Modelling and Validation of Concurrent Systems*. Springer-Verlag, 2009.

[JLL77] N. D. Jones, L. H. Landweber, and Y. E. Lien. Complexity of some problems in Petri nets. *Theoretical Computer Science*, 4:277–299, 1977.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, USA, 1972.

[KB09] Michael Köhler-Bußmeier. Hornets: Nets within nets combined with net algebra. In Karsten Wolf and Giuliana Franceschinis, editors, *Application and Theory of Petri Nets*, volume 5606 of *Lecture Notes in Computer Science*, pages 243–262. Springer-Verlag, 2009.

[KB11] Michael Köhler-Bußmeier. Decidability results for elementary object systems. Report of the department of informatics, Universität Hamburg, Fachbereich Informatik, 2011.

[KB12a] Michael Köhler-Bußmeier. On the complexity of safe, elementary Hornets. In Louchka Popova-Zeugmann, editor, *Proceedings of the 21st International Workshop on Concurrency, Specification, and Programming (CS&P 2012)*, volume 928, pages 203–214. CEUR Workshop Proceedings, 2012.

[KB12b] Michael Köhler-Bußmeier. Private communication, 2012.

[KBH09] Michael Köhler-Bußmeier and Frank Heitmann. On the expressiveness of communication channels for object nets. *Fundamenta Informaticae*, 93(1-3):205–219, 2009.

[KBH10a] Michael Köhler-Bußmeier and Frank Heitmann. Complexity of LTL model-checking for safe object nets. In B. Farwer, editor, *Proceedings of the International Workshop on Logic, Agents, and Mobility (LAM 2010)*, 2010.

[KBH10b] Michael Köhler-Bußmeier and Frank Heitmann. Safeness for object nets. *Fundamenta Informaticae*, 101(1-2):29–43, 2010.

[KBH11a] Michael Köhler-Bußmeier and Frank Heitmann. Liveness and reachability for elementary object systems. In Marcin Szczuka, Ludwik Czaja, Andrzej Skowron, and Magdalena Kacprzak, editors, *Concurrency, Specification and Programming (CS&P 2011), Proceedings*, Pułtusk, Poland, 2011. Białystok University of Technology.

[KBH11b] Michael Köhler-Bußmeier and Frank Heitmann. Liveness of safe object nets. *Fundamenta Informaticae*, 112(1):73–87, 2011.

[KBH12] Michael Köhler-Bußmeier and Frank Heitmann. Conservative elementary object systems. *Fundamenta Informaticae*, 120(3–4):325–339, 2012.

[KF07] Michael Köhler and Berndt Farwer. Object nets for mobility. In Jetty Kleijn and Alex Yakovlev, editors, *Petri Nets and Other Models of Concurrency - ICATPN 2007. 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2007, Siedlce, Poland, June 25-29, 2007. Proceedings*, volume 4546 of *Lecture Notes in Computer Science*, pages 244–262. Springer-Verlag, 2007.

[KM69] R. M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.

[KMR03] Michael Köhler, Daniel Moldt, and Heiko Rölke. Modelling mobility and mobile agents using nets within nets. In W. v. d. Aalst and E. Best, editors, *Application and Theory of Petri Nets*, volume 2679 of *Lecture Notes in Computer Science*, pages 121–140. Springer-Verlag, 2003.

[Köh04] Michael Köhler. *Objektnetze: Definition und Eigenschaften*, volume 1 of *Agent Technology – Theory and Applications*. Logos Verlag, Berlin, 2004.

[Köh07] Michael Köhler. The reachability problem for object nets. *Fundamenta Informaticae*, 79(3-4):401 – 413, 2007.

[Kos82] S.R. Kosaraju. Decidability of reachability in vector addition systems. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing (STOC'82)*, pages 267–281. ACM, 1982.

[KR03a] Michael Köhler and Heiko Rölke. Concurrency in mobile object-net systems. *Fundamenta Informaticae*, 54(2-3):221–235, 2003.

[KR03b] Michael Köhler and Heiko Rölke. Modelling sandboxes for mobile agents using nets within nets. In N. Busi and F. Martinelli, editors, *Workshop on Issues in Security and Petri Nets (WISP'03) at the International Conference on Application and Theory of Petri Nets 2003*. University of Eindhoven, 2003.

[KR04] Michael Köhler and Heiko Rölke. Properties of Object Petri Nets. In J. Cortadella and W. Reisig, editors, *Application and Theory of Petri Nets*, volume 3099 of *Lecture Notes in Computer Science*, pages 278–297. Springer-Verlag, 2004.

[KR05] Michael Köhler and Heiko Rölke. Reference and value semantics are equivalent for ordinary object Petri nets. In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets*, volume 3536 of *Lecture Notes in Computer Science*, pages 309–328. Springer-Verlag, 2005.

[Kri63] Saul A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[Kum00] Olaf Kummer. Undecidability in object-oriented Petri nets. *Petri Net Newsletter*, 59:18–23, 2000.

[Kum02] Olaf Kummer. *Referenznetze*. Logos Verlag, Berlin, 2002.

[Lak05] Charles Lakos. A Petri net view of mobility. In *Formal Techniques for Networked and Distributed Systems (FORTE 2005)*, volume 3731 of *Lecture Notes in Computer Science*, pages 174–188. Springer-Verlag, 2005.

[Lam92] J.L. Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99(1):79–104, 1992.

[Lip76] R. J. Lipton. The reachability problem requires exponential space. Research report 62, Department of Computer Science, Yale University, 1976.

[Lom00] Irina A. Lomazova. Nested Petri nets – a formalism for specification of multi-agent distributed systems. *Fundamenta Informaticae*, 43(1-4):195–214, 2000.

[Lom08] Irina A. Lomazova. Nested Petri nets for adaptive process modeling. In *Pillars of Computer Science. Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, volume 4800 of *Lecture Notes in Computer Science*, pages 460–474. Springer-Verlag, 2008.

[LP85] Orna Lichtenstein and Amir Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In Mary S. Van Deusen, Zvi Galil, and Brian K. Reid, editors, *Proceedings 12th Annual ACM Symposium on Principles of Programming Languages (POPL'85), New Orleans, Louisiana, USA, January 1985*, pages 97–107. ACM Press, 1985.

[LvHO+06] Irina A. Lomazova, Kees M. van Hee, Olivia Oanea, Alexander Serebrenik, Natalia Sidorova, and Marc Voorhoeve. Nested nets for adaptive systems. In *Petri Nets and Other Models of Concurrency - ICATPN 2006. 27th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, Turku, Finland, June 26-30, 2006. Proceedings*, volume 4024 of *Lecture Notes in Computer Science*, pages 241–260. Springer-Verlag, 2006.

[Ma05] Lu Ma. *A formal framework of a secure mobile agent system based on extended elementary object system*. PhD thesis, University of Illinois at Chicago, Chicago, IL, USA, 2005.

[May81] Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the 13th Annual Symposium on the Theory of Computing (STOC'81)*, pages 238–246. ACM, 1981.

[May84] Ernst W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM J. Comput.*, 13(3):441–460, 1984.

[Mil99] Robin Milner. *Communicating and mobile systems - the Pi-calculus.* Cambridge University Press, 1999.

[Mil01] Robin Milner. The flux of interaction. In J.-M. Colom and M. Koutny, editors, *Application and Theory of Petri Nets*, volume 2075 of *Lecture Notes in Computer Science*, pages 19–22. Springer-Verlag, 2001.

[MK96] T. Miyamoto and S. Kumagai. A multi agent net model of autonomous distributed systems. In *Proceedings. CESA'96, Symposium on Discrete Events and Manufacturing Systems*, pages 619–623, 1996.

[MT06] Lu Ma and Jeffrey J. P. Tsai. *Security modeling and analysis of mobile agent systems*, volume 5 of *Series in Electrical and Computer Engineering.* Imperial College Press, London, 2006.

[MT08] Lu Ma and Jeffrey J. P. Tsai. Formal modeling and analysis of a secure mobile-agent system. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 38(1):180–196, 2008.

[MTM04] Lu Ma, Jeffrey J. P. Tsai, and Tadao Murata. A secure mobile agent system model based on extended elementary object system. In *28th International Computer Software and Applications Conference (COMPSAC 2004), Design and Assessment of Trustworthy Software-Based Systems, 27-30 September 2004, Hong Kong, China, Proceedings*, pages 218–223. IEEE Computer Society, 2004.

[Pap94] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, MA, USA, 1994.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten.* PhD thesis, Technische Universität Darmstadt, 1962.

[Pet81] J. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall Inc., Englewood Cliffs NJ, 1981.

[Pnu81] Amir Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors, *Proceedings 5th International Symposium on Programming, Torino, Italy, April, 1982*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 1982.

[Rac78] C. Rackoff. The covering and boundedness problem for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978.

[Rei08] Klaus Reinhardt. Reachability in Petri nets with inhibitor arcs. In *Proceedings of the Second Workshop on Reachability Problems in Computations Models (RP 2008)*, volume 223 of *Electronic Notes in Theoretical Computer Science*, pages 239–264. Elsevier, 2008.

[Rot05] Jörg Rothe. *Complexity Theory and Cryptology.* Springer-Verlag, Berlin, Deutschland, 2005.

[RR98] Wolfgang Reisig and Grzegorz Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science.* Springer-Verlag, 1998.

[RVFE08] Fernando Rosa-Velardo and David de Frutos-Escrig. Name creation vs. replication in Petri net systems. *Fundamenta Informaticae*, 88:329–356, 2008.

[Sav70] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[Sch12] Laura Schmelter. Zum Erreichbarkeitsproblem in azyklischen Objektnetzen (in German). Bachelorarbeit, Universität Hamburg, Department Informatik, Vogt-Kölln Str. 30, D-22527 Hamburg, 2012.

[Sip97] Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, Boston, MA, USA, 1997.

[Ste95] I. A. Stewart. Reachability in some classes of acyclic Petri nets. *Fundamenta Informaticae*, 23:91–100, 1995.

[Tur36] A. M. Turing. On computable numbers with an application to the Entscheidungsproblem. *Proc. London Math. Society*, 2(42):230–265, 1936.

[Val91] Rüdiger Valk. Modelling concurrency by task/flow EN systems. In *3rd Workshop on Concurrency and Compositionality*, number

191 in GMD-Studien, St. Augustin, Bonn, 1991. Gesellschaft für Mathematik und Datenverarbeitung.

[Val98] Rüdiger Valk. Petri nets as token objects: An introduction to elementary object nets. In Jörg Desel and Manuel Silva, editors, *Application and Theory of Petri Nets*, volume 1420 of *Lecture Notes in Computer Science*, pages 1–25. Springer-Verlag, 1998.

[Val04] Rüdiger Valk. Object Petri nets: Using the nets-within-nets paradigm. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets. Advances in Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 819–848. Springer-Verlag, 2004.

[Var96] Moshe Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller and G. Birtwistle, editors, *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag, 1996.

[VW86] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings 1st IEEE Symposium on Logic in Computer Science (LICS'86), June 1986, Cambridge, Massachusetts, USA*, pages 332–344. IEEE Computer Society, 1986.

[Zil01] Silvano Dal Zilio. Mobile processes: A commented bibliography. In Franck Cassez, Claude Jard, Brigitte Rozoy, and Mark Dermot Ryan, editors, *Modeling and Verification of Parallel Processes. 4th Summer School, MOVEP 2000 Nantes, France, June 19-23, 2000. Revised Tutorial Lectures*, volume 2067 of *Lecture Notes in Computer Science*, pages 206–222. Springer-Verlag, 2001.