HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports
No. **14**; September 1990
ISSN 0783–5396
ISBN 951–22–0386–3

A COMPARATIVE STUDY OF METHODS FOR EFFICIENT REACHABILITY ANALYSIS

Marko Rauhamaa

Digital Systems Laboratory
Department of Computer Science
Helsinki University of Technology
Otaniemi, FINLAND

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports
No. **14**; September 1990

ISSN 0783–5396
ISBN 951–22–0386–3

A Comparative Study of Methods for Efficient Reachability Analysis

Marko Rauhamaa
M.Sc.

Abstract: Six recently proposed methods for efficient reachability analysis of distributed systems are studied and compared: symmetry method (Huber et al), parameterising method (Lindqvist), stubborn set method (Valmari), reduction theory (Berthelot, Haddad), symbolic model checking (Burch et al) and incomplete state-space generation (Holzmann). Petri systems are used as the main presentation formalism.

The methods are evaluated by how well they are suited for practical reachability graph generation by computer. One can summarise in brief that the stubborn set method, reduction theory and incomplete state-space generation are found very promising and they provide useful aid to practical reachability analysis. The symmetry method yields a significant reduction in memory space but hardly in generation time. Symbolic model checking is a very general method and immense sample state-spaces have been analysed with it but it is likely that for most practical systems the method performs far worse than the conventional methods. The parameterising method is impractical and also its theoretical profits are questionable.

Finally, it is observed that most of the methods are mutually independent and it is considered how they can be combined to complement each other. Indeed, the symmetry method, stubborn set method, reduction theory and incomplete state-space generation can all be used simultaneously to obtain better results than with any of them alone.

Keywords: reachability analysis, distributed systems, Petri systems.

Printing: TKK Monistamo; Otaniemi 1994

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

Phone: $\frac{90}{+358\text{–}0}$ 4511
Telex: 125 161 htkk fi
Telefax: +358–0–465 077
E-mail: lab@saturn.hut.fi

Contents

Preface iii

1 Introduction 1

1.1 Reachability Analysis 1

1.2 Petri Systems 2

1.3 Dining Philosophers Problem 2

1.4 Classification of Methods for Reachability Analysis 4

1.5 Remark on the Treatment of the Studied Methods 5

2 Symmetry Method 8

2.1 Introduction 8

2.2 Compressed Reachability Graph 9

2.3 Construction of Symmetries 11

2.3.1 Coloured Systems 11

2.3.2 Partial Symmetries 11

2.3.3 Symmetrical Transformations 13

2.3.4 Symmetry 14

2.4 Application of Symmetry Method 16

2.5 Conclusion 19

3 Parameterised Reachability Analysis xx

3.1 Introduction xx

3.2 Predicate/Transition Systems xxii

3.3 Parameterised Reachability Graph Generation xxiii

3.4 Discussion xxv

3.5 Conclusion xxix

4 Stubborn Set Method **30**

4.1 Introduction 30

4.2 Definition of Stubborn Sets 32

4.2.1 Place/Transition Systems 32

4.2.2 The Idea behind Stubborn Sets 33

4.2.3 Stubborn Sets 35

4.2.4 Example: Dining Philosophers 36

4.3 Computation of Stubborn Sets 37

4.3.1 Candidate List Algorithm 38

4.3.2 Algorithm using Strongly Connected Components . . . 38

4.3.3 Other Algorithms 40

4.4 Conclusion 41

5 Reduction Theory **42**

5.1 Introduction 42

5.2 Reduction Rules 42

5.3 Example 47

5.4 Conclusion 50

6 Other Approaches **52**

6.1 Symbolic Model Checking 52

6.2 Incomplete Reachability Analysis 55

7 Conclusion **59**

7.1 Evaluation of the Methods 59

7.2 Combining Methods 60

References **61**

Preface

This research forms my licentiate's thesis. I wish to thank Professor Leo Ojala, the supervisor of the thesis, Johan Lilius, Abbas Moslemie, Ilkka Niemelä and Patric Östergård from the Digital Systems Laboratory of Helsinki University of Technology for discussions and numerous valuable ideas.

I want to thank PhD Antti Valmari for careful inspection of the chapter that treats his work on stubborn sets and for the due corrections.

I am grateful to my wife Anne for the understanding, care and love, with which she created an ideal working motivation.

The research was funded by the Academy of Finland.

Otaniemi, 11th September 1990

Marko Rauhamaa

1 Introduction

1.1 Reachability Analysis

Telecommunication protocols, embedded systems and object oriented programmes are examples of *parallel and distributed systems*. They do not have one single central processing unit, but the control is distributed to several communicating processes or processors.

The operation of a distributed system is mostly too complex for a human mind to fully comprehend. Even a small system may have such a multitude of possible action interleavings that undesirable system states and erroneous operation sequences are introduced.

The conventional ways of verifying a sequential computer programme are testing and tracing ("debugging"). They are, however, not very well suited to distributed systems for several reasons. First, for a given input data or environment, a distributed system does not have but one possible operation sequence, so a single successful run cannot guarantee correct operation even in exactly the same circumstances. Secondly, the choice between the different operation sequences is often random, so the found errors may be hard to repeat and locate. Third, there are so many possible executions that it is difficult or unfeasible to construct a sufficiently covering test suite.

Reachability analysis is a suitable method of verification for distributed systems. It is based on generating the *reachability graph* for a formal abstract model of a system. The vertices of a reachability graph are all the states reachable from the initial state of the system and the edges are the actions leading from one state to another. If the system's state-space is finite, the reachability graph generation can be performed automatically by a computer.

Unfortunately, most practical systems have a finite but intractably large reachability graph. This is a little due to poor design conventions but mainly to the seemingly unavoidable combinatorial explosion. The size of the state-space may grow more than exponentially with respect to the size of a system configuration. Therefore an analyst is often limited to work on partial or incorrect models.

One way to "solve" the combinatorial problem is to perform partial reachability analysis. It is analogous to testing but the use of formal models makes it possible to automate the test generation, which yields faster and more covering analysis. This is not, however, the only possibility. Reachability graphs mostly contain repeated patterns which are caused by regularities or numerous different but equivalent operation alternatives of the model. This has led to development of various methods for more "intelligent" ways of generating the reachability graph.

Some of these methods are collected in this report. They are introduced and then judged by how well they are suited for practical reachability graph generation by computer. Finally, it is discussed how combinations of the methods can be formed.

1.2 Petri Systems

The modelling formalism chosen for this presentation is the Petri systems, though most of the methods are not dependent on this choice. Petri systems, however, are a simple formalism with very few primitives and they do not assume any specific application area. The knowledge of Petri systems is taken for granted in the report. Introductions to the formalism are, eg, [16] and [4, 5].

In this report, the semantics for the Petri systems is defined by a reachability graph with single transition steps. That is, interleaving semantics is adopted. The partial order semantics, which is often considered to be nearer the "reality", would introduce technical complexities but hardly afford any advantage to the treatment of the methods in this report.

Elementary net systems, place/transitions systems, coloured systems and predicate/transition systems are employed in defining the methods according to which of the several Petri system classes are used in the original papers. If one formalism had to be chosen, predicate/transition systems would perhaps be the best candidate. The formal high-level machinery is, however, unnecessarily heavy for presenting some of the methods. Nevertheless, the examples are mostly presented in a high-level formalism.

1.3 Dining Philosophers Problem

Throughout the chapters of this presentation, the *dining philosophers problem* is used as a graphical example for the application of most of the methods. In the theory of protocol verification, it has become one of the "canonical" examples and is in some form used as an example in many of the original papers.

Here is the version used in this report:

> Around a table, there are some philosophers sitting and thinking. Each of the philosophers will, of course, eventually become

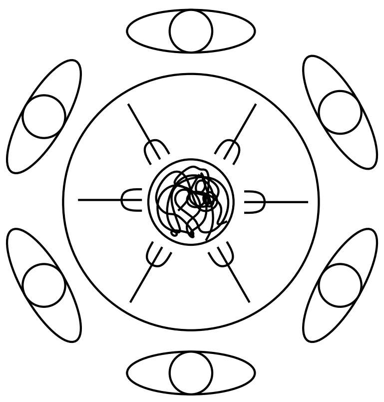


Figure 1: The system of six philosophers in the initial state.

> hungry. Therefore there is a large bowl of spaghetti in the center of the table and between every two philosophers, there is a fork.
>
> However clever the philosophers are as thinkers, they have little of applicative sense. Thus for them it is necessary to have two forks to be able to eat spaghetti, so they cannot all eat simultaneously. The problem is what kind of simple instructions the philosophers ought to be given according to which thinking and eating alternate smoothly.
>
> It is assumed that each philosopher has access only to the fork to his left and the one to his right. Also, he may operate only one fork at a time.

The solution used in the examples is the following incorrect protocol, which every philosopher obeys:

1. Whenever you are thinking and beginning to feel hungry, take the fork to your left as soon as it is on the table.
2. Take then the fork to your right as soon as it is on the table.
3. Eat.
4. When you are satisfied, put the fork in your left hand back on the table.
5. Put then the fork in your right hand on the table.
6. Go on thinking.

The dining philosophers problem seems to be ideal touchstone for the methods in question. It has an state-space with size exponential to the number of philosophers. The philosophers have a very intermingled competition about the resources. And yet the protocol has an apparent regularity which one should be able to utilise.

All of the methods that are tested against the dining philosophers seem to function well although some of them have serious drawbacks. It should be clear that one example cannot wholly judge the applicability and worth of a method.

1.4 Classification of Methods for Reachability Analysis

Gerard Holzmann has presented four types of state-space generation algorithms [9, 10], which, according to him, cover the field of "automated verification of the logical consistency of data communication protocols".

Type 1, Exhaustive Search. This is the obvious solution. There one, starting from the initial system state, constructs the whole state-space. All states have to be kept in memory in order to see which of the generated states are new and which have been generated before. Exhaustive search is limited by time and especially space limits.

Type 2, Partial Search. Here one does not necessarily examine all followers or a state but selects a subset of them according to some principle. The principle has an essential though difficultly predictable effect on, eg, what proportion of the errors are found.

Type 3, Stack Search. Only one path is stored in memory. The finiteness of the algorithm is accomplished by detecting the loops of the path (depth-first algorithm). A memory space that is roughly logarithmic to that of Type 1 is needed but exponentially more time is spent.

Type 4, Memoryless Search. No memory is required. One follows a single trace through the reachability graph. Some heuristics could be used. This method is in fact a form of testing.

Of the six methods reviewed in this report three can in some way be seen as fitting to Type 2: the symmetry method, the stubborn set method and Holzmann's incomplete reachability analysis. The other methods could not be classified to any of the four types. The types are, indeed, subcategories of a class of "naive" reachability graph generation methods. Therefore it would not be appropriate to classify into them all possible methods of automated verification.

The methods presented in this report can be characterised by a classification consisting of three attributes.

1. How much of the work needed in the method is or can be done beforehand? How complex computation is needed at every state of the state-space generation?

2. How dependent is the method on some external knowledge about the system? Or can the method be applied to the sheer formal model? Is some kind of heuristic necessary?

3. How "naive" or "intelligent" is the method? How high-level properties or regularities of the system are utilised?

Holzmann can give a categorical judgement about the practicality of each of the four types, whereas the three attributes do not determine the goodness of a method. For instance, while ability to do work beforehand is usually inexpensive, a dynamic method may, on the other hand, yield greater reduction in generation time. While external knowledge about the system may enable the use of an efficient method, it may not be known how generally the necessary information is available, depending on the expertise of the analyst or the special character of the system. And while an intelligent method may produce great savings in the necessary memory space, a straightforward method may generate far more states in the same time.

Table 1 presents a classification of the methods treated in this report. The following chapters help to form an opinion of how good the methods are from different points of view. Chapter 7 presents an estimate of the practical value of the methods and discusses how the presented methods could complement each other.

1.5 Remark on the Treatment of the Studied Methods

The level of detail in different chapters varies considerably. Some present exact formal definitions and complete proofs (Chapters 2 and 4), while others seek to give a good intuition by semiformal descriptions and examples. This inconsequence is motivated below.

The symmetry method is originally presented in a seemingly formal way [11]. The exactness does not, however, extend down to the roots of the definitions. Since the low level is not obvious and it is naturally needed in the application of the method, it is defined in detail in Chapter 2. The definitions have also been formulated slightly differently and they have been generalised in some respects, so the proofs have been given, too. Unlike the original presentation, Chapter 2 tries to separate the idea of the method and its presented instance. This leads to far shorter proofs than in the report of Huber et al.

Table 1: A classification of the methods for reachability analysis.

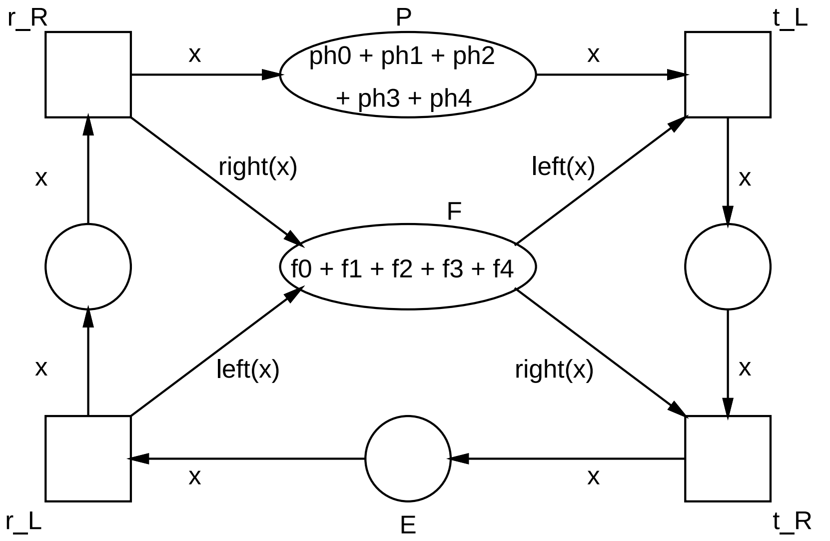
Ch	Method		Attribute
2	Symmetry method	1	Complex (≈ exponential) calculations are needed in every state.
		2	The symmetry relation must be given by the analyst but its soundness may be verified automatically.
		3	The regular structure of the system is utilised efficiently, but only some perfect regularities can be exploited.
3	Parameterising method	1	Complex (≈ exponential) calculations are needed in every state.
		2	No external knowledge is needed.
		3	Many system properties and regularities seem to be disregarded or even lost.
4	Stubborn set method	1	Depending on the algorithm, either small (linear) or tolerable (quadratic or worse) computation is performed in every state.
		2	No external knowledge is necessary but it may be useful.
		3	Utilises subtle dynamic system dependencies in a controlled way.
5	Reduction theory	1	All work is done prior to the ordinary reachability graph generation.
		2	External knowledge may or may not be necessary depending on the applied reduction rules. It is always useful.
		3	The method may be considered “semi-intelligent”. The system model is mechanically matched to a fixed set of structural rules, which are designed to take advantage of the regularities of the system.
6	Symbolic model checking	1	Depending on the case, the work in every state may be from negligible to infeasibly complex.
		2	No external knowledge is necessary.
		3	In principle, symbolic logic could schematise most complex properties of a system.
6	Incomplete reachability analysis	1	Work in every state is far smaller (≈ two orders of magnitude) than in the ordinary reachability graph generation.
		2	No external knowledge is necessary.
		3	Does not utilise the specific system properties in any manner.

Lindqvist's parameterisation method is almost merely sketched in Chapter 3. It is due to the extremely complex formalism and definitions of the references [12, 13] and the unfortunate fact that neither the practical method nor its theoretical merits seem to be very successful.

Valmari's stubborn set method is an interesting new generalisation of the various priority methods. However, the intelligibility of his papers [18, 19, 20, 21, 22, 23] is probably not the best possible. Chapter 4 tries to highlight the simple but clever principles of the new concepts, semistubbornness and stubbornness, and then explain the complex definition of structural semistubbornness. This kind of two-level treatment is analogous to that of Chapter 2.

The reduction theory is too large to be thoroughly handled in one chapter (Chapter 5). The general ideas are described and the reader is referred to some of the main sources [1, 2, 8, 3, 7].

The approaches of Chapter 6 are of a little different nature than the others, so they are outlined more briefly. However, Holzmann's method (Section 6.2) is so simple that even the short description gives enough information, eg, for implementing the method as a computer programme.



$$\begin{array}{rcl} \mathit{left}(\phi_i) & = & f_i \\ \mathit{right}(\phi_i) & = & f_{(i+1) \bmod 5} \end{array}$$

Figure 2: A coloured system model of the dining philosophers.

2 Symmetry Method

2.1 Introduction

In solving mathematical problems, it is often possible to reduce the necessary effort by detecting and utilising symmetries. Also in system modelling, it is usual to meet different kinds of symmetrical structures that seem to lead to a severe redundancy in the construction of the reachability graph. A research group consisting of Peter Huber, Arne M. Jensen, Leif O. Jepsen and Kurt Jensen has presented a method that formalises some types of symmetries in Petri systems [11] so that symmetrical markings are represented by a single vertex and symmetrical transitions by a single edge in the compressed reachability graph.

Figure 2 represents a coloured system model of the dining philosophers. When constructing the reachability graph, one immediately meets with a marking where only the philosopher φ_0 is eating and the others thinking and also another marking where it is the philosopher φ_1 who is eating. Normally, they are considered different markings, so the reachability graph consists of several symmetrical subgraphs. Using the symmetry method, however, one marking can represent an equivalence class of markings.

The philosopher model contains two of the five defined symmetry types. The basic symmetry type is a shift or rotation by which the bags $\langle\varphi_0\rangle + \langle\varphi_1\rangle$ and $\langle\varphi_1\rangle + \langle\varphi_2\rangle$ are symmetrical while $\langle\varphi_0\rangle + \langle\varphi_2\rangle$ is not symmetrical to the others. The forks are of a related symmetry type: f_0 is symmetrical to f_1 in the same way as φ_0 is symmetrical to φ_1.

2.2 Compressed Reachability Graph

Consider a P/T system $\langle P, T, W, M_0\rangle$ (the definition of P/T systems will be presented in Section 4.2.1). The reachability graph $\langle V, E\rangle$ of the system consists of markings and transition instances such that $V \subseteq \mathcal{M}$ and $E \subseteq \mathcal{M} \times T \times \mathcal{M}$, where $\mathcal{M} = N^P$ is the set of possible markings. The source and target (or destination) of an edge are denoted by the functions

$$s : \mathcal{M} \times T \times \mathcal{M} \to \mathcal{M} : s(M, t, M') = M$$

and

$$d : \mathcal{M} \times T \times \mathcal{M} \to \mathcal{M} : d(M, t, M') = M'$$

respectively.

Suppose there exists an equivalence relation $\sim$ called symmetry between the potential vertices (or markings) and on the other hand between the potential edges of the reachability graph. The equivalence class generated by a is denoted by $[a] = \{x \mid x \sim a\}$. The symmetry between markings must be related to that between edges in such a way that for any $\langle M, t, M'\rangle \in \mathcal{M} \times T \times \mathcal{M}$,

$$\{s(e) \mid e \in [\langle M, t, M'\rangle]\} = [M] \tag{1}$$

and

$$\{d(e) \mid e \in [\langle M, t, M'\rangle]\} = [M']. \tag{2}$$

Then a compressed reachability graph can be constructed as described in the following definition.

Definition 2.1 *Let $\sim$ be a symmetry. The* compressed reachability graph *of a P/T system is the pair $\langle \mathcal{V}, \mathcal{E}\rangle$ with $\mathcal{V} \subseteq 2^{\mathcal{M}}$, $\mathcal{E} \subseteq 2^{\mathcal{M}} \times 2^{\mathcal{M}\times T\times\mathcal{M}} \times 2^{\mathcal{M}}$ such that the following conditions are satisfied:*

1. *$[M_0] \in \mathcal{V}$.*
2. *If $[M] \in \mathcal{V}$ and $M[t\rangle M'$, then $[M'] \in \mathcal{V}$ and $\langle [M], [\langle M, t, M'\rangle], [M']\rangle \in E$.*
3. *$\mathcal{V}$ and $\mathcal{E}$ have no other elements.*

Depending on the choice of the symmetry $\sim$, the compressed reachability graph may or may not be equivalent to the ordinary reachability graph. In order that the kind of correspondence could be decided, the "unfolding" of a compressed reachability graph is defined.

Definition 2.2 *The unfolded reachability graph of a compressed reachability graph* $\langle \mathcal{V}, \mathcal{E} \rangle$ *is*

$$U(\langle \mathcal{V}, \mathcal{E} \rangle) = \langle V_U, E_U \rangle$$

where $V_U = \bigcup \mathcal{V}$ *and* $E_U = \bigcup \{ \langle M, t, M' \rangle \mid \langle [M], [\langle M, t, M' \rangle], [M'] \rangle \in \mathcal{E} \}$.

If the unfolded graph is equal to the ordinary reachability graph, then the compressed reachability graph is but a more compact representation of the same system behaviour and it naturally has exactly the same system states and properties.

Theorem 2.3 *The reachability graph* $\langle V, E \rangle$ *is equivalent to the compressed reachability graph* $\langle \mathcal{V}, \mathcal{E} \rangle$*, ie,* $\langle V, E \rangle = U(\langle \mathcal{V}, \mathcal{E} \rangle)$*, if*

$$[M_0] = \{M_0\} \tag{3}$$

and for all $\langle M_1, t_1, M_1' \rangle, \langle M_2, t_2, M_2' \rangle \in \mathcal{M} \times T \times \mathcal{M}$,

$$\langle M_1, t_1, M_1' \rangle \sim \langle M_2, t_2, M_2' \rangle \Rightarrow (M_1[t_1\rangle M_1' \Leftrightarrow M_2[t_2\rangle M_2'). \tag{4}$$

Proof: First, it will be proved by induction that $V_U \subseteq V$ and $E_U \subseteq E$.

By Def 2.1, $[M_0] \in \mathcal{V}$. By Def 2.2 and (3), $[M_0] = \{M_0\} \subseteq V_U$ or $M_0 \in V_U$. Also by Def 4.3, $M_0 \in V$.

Suppose $[M] \in \mathcal{V}$ and $M[t\rangle M'$. By Def 2.2 and the inductive assumption, $[M] \subseteq V_U$ and $[M] \subseteq V$. Then by Def 2.1, $[M'] \in \mathcal{V}$ and $\langle [M], [\langle M, t, M' \rangle], [M'] \rangle \in \mathcal{E}$, and by Def 2.2, $[M'] \subseteq V_U$ and $[\langle M, t, M' \rangle] \subseteq E_U$. But since $M[t\rangle M'$, by (4), for all $\langle M_1, t_1, M_1' \rangle \in [\langle M, t, M' \rangle]$, $M_1[t\rangle M_1'$, so by (1), (2) and Def 4.3, $[M'] \subseteq V$ and $[\langle M, t, M' \rangle] \subseteq E$. Thus $V_U \subseteq V$ and $E_U \subseteq E$.

Then it will be proved that $V \subseteq V_U$ and $E \subseteq E_U$.

As seen above, $M_0 \in V$ and $[M_0] = \{M_0\} \subseteq V_U$.

Suppose $M \in V$ and $M[t\rangle M'$. By Def 4.3, $M' \in V$ and $\langle M, t, M' \rangle \in E$. According to the inductive assumption, $[M] \in \mathcal{V}$, and by Def 2.1, $[M'] \in \mathcal{V}$ and $\langle [M], [\langle M, t, M' \rangle], [M'] \rangle \in \mathcal{E}$. Thus by Def 2.2, $[M'] \subseteq V_U$ and $[\langle M, t, M' \rangle] \subseteq E_U$, so $M' \in V_U$ and $\langle M, t, M' \rangle \in E_U$. It follows that $V \subseteq V_U$ and $E \subseteq E_U$.

Thus it has been proved that $V = V_U$ and $E = E_U$. $\square$

Even if (3) and (4) were not satisfied, the compressed reachability graph may have properties in common with the ordinary graph. In the proof of Theorem 2.3, conditions (3) and (4) are not used in proving that $V \subseteq V_U$ and $E \subseteq E_U$. Thus if there are deadlocks in $\langle \mathcal{V}, \mathcal{E} \rangle$, they are there in $\langle V, U \rangle$, and all the token count upper bounds of $\langle \mathcal{V}, \mathcal{E} \rangle$ apply to $\langle V, U \rangle$, too.

The symmetry is called *sound* if (4) holds. If the symmetry is sound but the initial marking is not symmetrical (ie, (3) does not hold), the liveness properties of $\langle V, E \rangle$ and $\langle V_U, E_U \rangle$ are still identical. For let $M \in V$ be a dead marking. Then $[M] \in \mathcal{V}$. Since no transition is enabled in M, none is enabled in any of $[M]$ because of (1) and (4). Thus every marking in $[M]$ is dead even if it is not there in the ordinary reachability graph. The same applies to boundedness, too, if in all symmetry classes every marking has the same token count upper bound. This is, indeed, the case when symmetries are constructed as shown in the next section.

2.3 Construction of Symmetries

The definition of a symmetry gave no suggestion about how to construct symmetries. In [11], Huber et al define three symmetry types that are fairly easy for the analyst to detect in a model and which form a basis for a class of symmetries. The necessary definitions are complex but not very difficult to understand. They are given in detail in order to reduce the trouble of those who would like to apply the method. The authors give the definitions for coloured systems, which are here defined to be a special case of place/transition (P/T) systems.

2.3.1 Coloured Systems

Definition 2.4 *Let $\mathcal{P}$ be a set of place labels, $\mathcal{T}$ be a set of transition labels ($\mathcal{P} \cap \mathcal{T} = \emptyset$) and $\mathcal{C}$ be a set of colours. Then a* coloured system *is a P/T system $\langle P, T, W, M_0 \rangle$ with $P \subseteq \mathcal{P} \times \mathcal{C}$ and $T \subseteq \mathcal{T} \times \mathcal{C}$.*

In the sequel, one refers to the coloured system $\langle P, T, W, M_0 \rangle$ and $\mathcal{P}$, $\mathcal{T}$ and $\mathcal{C}$ are as above. The colouring function C is defined to be

$$C : \mathcal{P} \cup \mathcal{T} \to 2^{\mathcal{C}} : C(x) = \{c \in \mathcal{C} \mid \langle x, c \rangle \in P \cup T\}.$$

2.3.2 Partial Symmetries

Let Q be a collection of colour sets so that

$$\{C(x) \mid x \in \mathcal{P} \cup \mathcal{T}\} \subseteq Q \subseteq 2^{\mathcal{C}}.$$

Each element of Q is regarded as a *partial symmetry*. These symmetry types constitute the symmetry relation $\sim$. (There may be two colour sets that have a common component, say, $C(x) = A \times B$ and $C(y) = B \times D$, even if the component is not a colour set of any $z \in \mathcal{P} \cup \mathcal{T}$. This is the reason for the fact that Q may contain elements other than the $C(x)$, like B in the example.) The set Q is partitioned into five sets according to the *partial symmetry types*: Q_i (identity), Q_s (shift), Q_p (permutation), Q_r (related) and Q_π (product).

With some of the partial symmetry types, additional information is associated. Each $q \in Q_s$ has a *distance operation* $-_q$ that fulfils the following axioms (all quantifiers range over q):

$$\forall a \forall b \forall c \exists x \; a - b = c - x \tag{5}$$
$$\forall a \forall b \forall c \forall d \, (a - b = c - d \Rightarrow d - b = c - a) \tag{6}$$

from which following properties are obtained:

$$\forall a \forall b \forall c \exists x \; a - b = x - c \tag{7}$$
$$\forall a \forall b \forall c \forall d \, (a - b = c - d \Rightarrow b - a = d - c) \tag{8}$$
$$\forall a \forall b \forall c \exists x \, (x - a = c - b \wedge b - a = c - x) \tag{9}$$
$$\forall a \forall b \; a - a = b - b \;\; [=_{\text{def}} 0] \tag{10}$$
$$\forall a \forall b \forall c \, (a - b = a - c \Leftrightarrow b - c = 0) \tag{11}$$
$$\forall a \forall b \forall c \, (a - c = b - c \Leftrightarrow a - b = 0) \tag{12}$$
$$\forall a \forall b \forall c \forall x \forall y \forall z \, (c - b = z - y \wedge b - a = y - x \Rightarrow c - a = z - x). \tag{13}$$

In order to simplify the constructions and proofs, it is also assumed that

$$\forall a \forall b \, (a - b = 0 \Rightarrow a = b). \tag{14}$$

The target set of the operation $-_q$ is not specified but it could be q itself or (mostly) some number or vector space (N, Z, Q, R, C, R^n, $\ldots$). If q is a finite set $\{0, \ldots, n-1\}$, then $-_q$ could be (but is not necessarily) the difference (mod n). If $q = Z$ or R, $-_q$ might be the ordinary difference between integer or real numbers.

The parent function $F : Q_r \to Q$ assigns each related colour set a parent set, and $f_q : q \overset{\text{bij}}{\to} F(q)$ bijectively relates each colour in a colour set q to a colour in q's parent set.

Each product colour set $q \in Q_\pi$ is a Cartesian product of a family $\{q_i\}_{I(q)}$ of colour sets. The projection functions $\Pi_q : I(q) \to Q$ and $\pi_q : q \times I(q) \to \mathcal{C}$ so that for all $c \in q$ and $i \in I(q)$, $\pi_q(c, i) \in \Pi_q(i)$.

It is assumed that the structure of Q is well-founded, ie, there are no loops formed by F or Π.

2.3.3 Symmetrical Transformations

The set Q of partial symmetries (with its structure) completely fixes a symmetry relation $\sim$. The relationship is defined by means of symmetrical transformations.

Definition 2.5 *Let ϕ be a function with Q as its domain such that*

$$\phi(q) : q \to \mathcal{C}$$

($\phi(q)$ is subsequently denoted by ϕ_q). Then ϕ is a symmetrical transformation, *if for any $q \in Q$,*

1. *if $q \in Q_i$, then ϕ does not transform q: $\forall c \in q$*

   $$\phi_q(c) = c,$$

2. *if $q \in Q_s$, then ϕ preserves distances in q: $\forall c \in q \, \forall c' \in q$*

   $$\phi_q(c) -_q \phi_q(c') = c -_q c';$$

3. *if $q \in Q_p$, then ϕ is bijective in q: $\forall c \in q \, \forall c' \in q$*

   $$\phi_q(c) = \phi_q(c') \Rightarrow c = c'$$

   and $\forall c \in q \, \exists c' \in q$

   $$c = \phi_q(c');$$

4. *if $q \in Q_r$, then ϕ acts in accordance with the relation between q and $F(q)$: $\forall c \in q$*

   $$f_q(\phi_q(c)) = \phi_{F(q)}(f_q(c));$$

   and

5. *if $q \in Q_\pi$, then ϕ acts in accordance with the Cartesian product: $\forall i \in I(q) \, \forall c \in q$*

   $$\pi_q(\phi_q(c), i) = \phi_{\Pi_q(i)}(\pi_q(c, i)).$$

The definition of a symmetrical transformation can now be augmented to include places, transitions and markings. If $\langle x, c \rangle \in P \cup T$, then $\phi(x, c) = \langle x, \phi_{C(x)}(c) \rangle$. If $M \in \mathcal{M}$ then $\phi(M)(p) = M(\phi^{-1}(p))$. (That ϕ has an inverse will be proved in the next subsection.)

2.3.4 Symmetry

Let Φ be the set of all symmetrical transformations defined by Q. Then Φ fixes a symmetry $\sim$ such that for any $M, M' \in \mathcal{M}$,

$$M \sim M' \text{ iff } \exists \phi \in \Phi \; M = \phi(M')$$

and for any $\langle M_1, t_1, M_1' \rangle, \langle M_2, t_2, M_2' \rangle \in \mathcal{M} \times T \times \mathcal{M}$,

$$\langle M_1, t_1, M_1' \rangle \sim \langle M_2, t_2, M_2' \rangle \text{ iff } \exists \phi \in \Phi (M_1 = \phi(M_2) \wedge t_1 = \phi(t_2) \wedge M_1' = \phi(M_2')).$$

It remains only to prove that $\sim$ is an equivalence relation.

Lemma 2.6 *The identity transformation $I : I_q(c) \equiv c$ is symmetrical.*

Proof: Each of the five conditions in Definition 2.5 are trivially satisfied when I is substituted for ϕ in them. □

Lemma 2.7 *For any symmetrical transformation ϕ, the inverse transformation ϕ^{-1} (for which $\phi_q(\psi_q(c)) = \psi_q(\phi_q(c)) = q$) exists and is symmetrical.*

Proof: Let ϕ be a symmetrical transformations. The inverse $\phi_q^{-1} = \psi_q$ is constructed as follows.

1. If $q \in Q_i$, then $\psi_q(c) = \phi_q(c) = c$.
2. If $q \in Q_s$, then $\psi_q(c)$ is chosen to be the colour in q fulfilling $\psi_q(c) -_q c = c -_q \psi_q(c)$, which is possible according to (5). Now $\phi_q(\psi_q(c)) -_q \phi_q(c) = \psi_q(c) -_q c$. So by (8) and the definition of ψ_q, $\phi_q(\psi_q(c)) -_q \phi_q(c) = c -_q \phi_q(c)$, and by (12) and (14), $\phi_q(\psi_q(c)) = c$. On the other hand, by the definition of ψ_q, $\phi_q(c) -_q \psi_q(\phi_q(c)) = \phi_q(\phi_q(c)) -_q \phi_q(c) = \phi_q(c) -_q c$. So by (13), $\psi_q(\phi_q(c)) = c$. Thus ψ_q is the inverse of ϕ_q. Its symmetry is obtained as follows: $\phi_q(c) -_q \phi_q(c') = c -_q c'$, so by (6), $c -_q \phi_q(c) = c' -_q \phi_q(c')$, or $\psi_q(c) -_q c = \psi_q(c') -_q c'$, whence $\psi_q(c) -_q \psi_q(c') = c -_q c'$.
3. Suppose $q \in Q_p$. Then ψ_q is chosen such that for all c, $c = \phi_q(\psi_q(c))$, which is possible by the surjectivity of ϕ_q. Then $\phi_q(\psi_q(\phi_q(c))) = \phi_q(c)$. But by the injectivity of ϕ_q, $\psi_q(\phi_q(c)) = c$, so ψ_q is the inverse of ϕ_q. It is also symmetrical: let $\psi_q(c) = \psi_q(c')$. By the surjectivity of ϕ_q, there are colours d and d' such that $\psi_q(\phi_q(d)) = \psi_q(\phi_q(d'))$, or $d = d'$, or $c = c'$. The injectivity of ψ_q follows immediately from the fact that $\psi_q(\phi_q(c)) = c$.

4. Suppose $q \in Q_r$. Then one can define $\psi_q(c) = f_q^{-1}(\phi_{F(q)}^{-1}(f_q(c)))$ assuming inductively that $\phi_{F(q)}^{-1}$ exists. If $\psi_q = \phi_q^{-1}$, it will automatically be symmetrical. Indeed, it is since $\phi_q(\psi_q(c)) = f_q^{-1}(\phi_{F(q)}(f_q(\psi_q(c)))) = f_q^{-1}(\phi_{F(q)}(\phi_{F(q)}^{-1}(f_q(c)))) = f_q^{-1}(f_q(c)) = c$, and $\psi_q(\phi_q(c)) = f_q^{-1}(\phi_{F(q)}^{-1}(f_q(\phi_q(c)))) = f_q^{-1}(\phi_{F(q)}^{-1}(\phi_{F(q)}(f_q(c)))) = f_q^{-1}(f_q(c)) = c$.

5. The case of $q \in Q_\pi$ is proved analogously to that of Q_r. The inverse of the Cartesian product is constructed from the inverses of the components.

□

Lemma 2.8 *For all symmetrical transformations ϕ and ψ, the composition transformation $\phi \circ \psi : (\phi_q \circ \phi_q)(c) = \phi_q(\psi_q(c))$ is symmetrical.*

Proof: Let ϕ and ψ be symmetrical transformations.

1. If $q \in Q_i$, then $\phi_q(\psi_q(c)) = \psi_q(c) = c$.

2. If $q \in Q_s$, then $\phi_q(\psi_q(c)) -_q \phi_q(\psi_q(c')) = \psi_q(c) -_q \psi_q(c') = c -_q c'$.

3. Suppose $q \in Q_p$. Then $\phi_q(\psi_q(c)) = \phi_q(\psi_q(c'))$ implies $\psi_q(c) = \psi_q(c')$ which implies $c = c'$. For any c there exists c' such that $c = \phi_q(c')$. For this c' there exists c'' such that $c' = \psi_q(c'')$. So $\forall c \in q\, \exists c'' \in q\; c = \phi_q(\psi_q(c''))$.

4. Suppose $q \in Q_r$. Then $f_q(\phi_q(\psi_q(c))) = \phi_{F(q)}(f_q(\psi_q(c))) = \phi_{F(q)}(\psi_{F(q)}(f_q(c)))$.

5. Suppose $q \in Q_\pi$. Then $\pi_q(\phi_q(\psi_q(c))) = \phi_{\Pi_q(i)}(\pi_q(\psi_q(c), i)) = \phi_{\Pi_q(i)}(\psi_{\Pi_q(i)}(\pi_q(c, i)))$.

□

By applying Lemmas 2.6 to 2.8 to the construction of $\sim$, the following theorem is immediately obtained.

Theorem 2.9 *If $\sim$ is constructed from the set Φ of all symmetrical transformations, then $\sim$ is equivalence relation.*

Finally, an important theorem is proved that enables one by structural inspection to become assured of the soundness of the symmetry (Condition (4)).

Theorem 2.10 *A symmetry $\sim$ constructed from the set Φ of symmetrical transformations is sound, iff for all $\phi \in \Phi$, $\langle x, y\rangle \in \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$, $c \in C(x)$ and $d \in C(y)$,*

$$W(x, c, y, d) = W(x, \phi_{C(x)}(c), y, \phi_{C(y)}(d)).$$

Proof: Let for all $\langle x, y\rangle \in P \times T \cup T \times P$, $W(x, y) = W(\phi(x), \phi(y))$ and $\langle M_1, t_1, M_1'\rangle \sim \langle M_2, t_2, M_2'\rangle$. Then there is $\phi \in \Phi$ such that $M_1 = \phi(M_2)$, $t_1 = \phi(t_2)$ and $M_1' = \phi(M_2')$. Transition t_1 is enabled iff $\forall p \in P\ M_1(p) \geq W(p, t_1)$ iff $\forall p \in P\ \phi(M_2)(p) \geq W(p, \phi(t_2))$ iff $\forall p \in P\ M_2(\phi^{-1}(p)) \geq W(p, \phi(t_2))$ iff $\forall p' \in P\ M_2(\phi^{-1}(\phi(p'))) \geq W(\phi(p'), \phi(t_2))$ iff $\forall p' \in P\ M_2(p') \geq W(\phi(p'), \phi(t_2))$ iff $\forall p' \in P\ M_2(p') \geq W(p', t_2)$ iff transition t_2 is enabled. It is similarly proved that $M_1[t_1\rangle M_1'$ iff $M_1[t_1\rangle M_1'$.

For the reverse direction, suppose for instance that $W(p, t) < W(\phi(p), \phi(t))$ for some $p \in P$, $t \in T$ and $\phi \in \Phi$. Then there exists a marking M such that $M(p') = W(p', t)$ for all places p'. Then t is enabled at M but $\phi(t)$ is not enabled at $\phi(M)$ for $\phi(M)(\phi(p)) = M(p) = W(p, t) < W(\phi(p), \phi(t))$. Thus $\sim$ is not sound. The other case $W(t, p) < W(\phi(t), \phi(p))$ is proved similarly. □

2.4 Application of Symmetry Method

When a practical algorithm implementing the symmetry method is considered, it can be observed that the systems under inspection, or at least the colour sets of shift or permutation symmetry types, are mostly finite. A shift in a finite colour set q is a rotation and fully determined by a cyclic successor function and a natural number from 0 to $|q| - 1$, so there are $|q|$ different shift functions $\phi(q, c)$. A permutation in a finite colour set q is one of the $|q|!$ possible shuffles.

The algorithm proposed by [11] stores one marking for each equivalence class of symmetrical markings. Starting from the initial marking, the reachability graph is constructed as usual, except when one comes across a marking that is symmetrical to a previously generated one. Then the edge is put to point the latter (probably commented by the transformation). This approach has two disadvantages. First, symmetrical transitions or edges may have several representatives. And secondly, which is more severe, every generated marking must be compared to each previously generated marking by means of the laborious symmetry test. Huber et al claim to be able to reduce the work "drastically" by using a hash coding, though they do not describe at all what the hash coding is like.

However, it seems possible to construct a normal form from a marking or a transition with a work equal to checking the symmetry once. Then the symmetry test between two markings is simply the equality test between the normal forms of the markings. The performance is at least as good as in the algorithm of Huber et al.

The symmetry check between two markings is the method's heel of Achilles. It is carried out as follows. Let the compared markings be M and M'. The following "algorithm" tries to fix the values for a symmetrical transformation ϕ.

For each $q \in Q$,
  initially the values of ϕ_q are not fixed,
  for each $x \in \mathcal{P}$ with $C(x) = q$,
    let $\forall n \in N\ C_n = \{c \in q \mid M(x, c) = n\}$,
    let $\forall n \in N\ C'_n = \{c \in q \mid M'(x, c) = n\}$,
    if $\exists n \in N\ |C_n| \neq |C'_n|$
      then M and M' are not symmetrical,
    for each $n \in N$ with $C_n \neq \emptyset$,
      fix values $\phi_q(c)$ yet unfixed for each $c \in C_n$
      or if it is not possible, try backtracking to the previous n and refixing ϕ_q
    or if it is not possible, try backtracking to the previous x and refixing ϕ_q
  or if it is not possible, try backtracking to the previous q and refixing ϕ_q
or if that is not possible, either, M and M' are not symmetrical.

Note that constructing the C_n corresponds to transforming the P/T system into the equivalent elementary net system.

The fixing of the $\phi_q(c)$ depends on the symmetry type of q.

1. If $q \in Q_i$ (identity), the value is already fixed to $\phi_q(c) = c$. If $C_n \neq C'_n$, the markings cannot be symmetrical. This case is, of course, computationally trivial.

2. If $q \in Q_s$ (shift), an $c \in C_n$ is associated with one $c' \in C'_n$. This immediately fixes ϕ_q so that for each $d \in C_n$, $\phi_q(d) -_q d = c' -_q c$. There are $|C_n| < |q|$ ways to construct the fixing. They are tried out one by one until $C'_n = \{\phi_q(c) \mid c \in C_n\}$. Sometimes there are several possible ways to fix ϕ_q successfully, which must be checked when backtracking.

   The computational performance is fair.

3. If $q \in Q_p$ (permutation), one would not like to traverse all the $|C_n|!$ different shuffles. Huber et al vaguely describe a "multiarc diagram calculus" which does not seem any more efficient than the procedure described here.

Each unfixed $\phi_q(c)$ are set to an unused $c' \in C'_n$. If previous fixings lead to an inconsistency, backtracking must take place. Better than going through all the possible permutations, the algorithm is still very hard to execute.

4. If $q \in Q_r$ (related), every element of C_n and C'_n are mapped with f_q to $F(q)$ and the procedure of $F(q)$ is applied.

   This and the next transformation are the reason for the need of backtracking to preceding q's in Q.

5. If $q \in Q_\pi$ (product), fixing a $\phi_q(c)$ is done by simultaneously fixing each of the $\phi_{\Pi_q(i)}(\pi_q(c, i))$.

   Huber et al deal with the product case differently. Let $q = q_1 \times \cdots \times q_k$. A place $\langle x, \langle c_1, \ldots, c_k\rangle\rangle$ is treated as if in the system there were places $\langle\langle x, 1\rangle, c_1\rangle, \ldots \langle\langle x, k\rangle, c_k\rangle$. These places can then be handled according to the component colour sets. This kind of transformation does not preserve the original structure. Therefore it is necessary to check the resulting ϕ with the markings M and M' at the end of the algorithm. A failure will cause backtracking. This does not seem a very good way to handle the product case, but Huber et al conjecture backtracking is seldom needed or there are only a few possibilities to go through.

Normal forms of markings can be generated in a way that resembles symmetry testing. The order of the $q \in Q$, $x \in \mathcal{P}$ and $n \in N$ must be the same during the whole generating process. Each $c \in q$ is given a weight value $w(c)$. The weight of a set $C_n \subset q$ is the sum $\sum_{c \in C_n} w(c)$. One first tries to fix a symmetrical transformation ϕ_q so that it minimises the weight of C_n for the first q, x and n. Then one goes on with the next q, x and n and so on until the whole marking is transformed into its "lightest".

If at a stage two fixings yield the same weight, the computation is split into several branches and the choice is left to the next stage. If two transformations produce the same weights, they should also result in the same marking. This seems to be automatically the case when the weights of the colours are chosen to be, eg, subsequent natural numbers or subsequent powers of two. What may prove to be a problem is that the permutation type causes $|C_n|!$ branches to the computation at the first stage. Therefore it is necessary to compute a stage breadth-first-like simultaneously in every branch in order to cut the branches as early as possible.

2.5 Conclusion

The symmetry method is easy to understand and can significantly cut down the size of the reachability graph. The "reduction coefficient" is at best $O(1)$ for each identity, $O(1/|q|)$ for each shift and $O(1/|q|!)$ for each permutation symmetry type. The method requires external knowledge about the system so that the partial symmetry types can be defined. However, this does not cause any difficulty since the analyst usually sees them immediately in the model.

The five symmetry types (identity, shift, permutation, related and product) are natural although they do not form any complete set. For instance, a power set symmetry type would be easy to define and could turn out useful in models of protocols with voting.

Unfortunately, the computation time may not decrease in the symmetry method at all since whatever is gained in the size of the reachability graph is lost in comparing the markings or in constructing the normal forms. Nevertheless, it may be quicker (by a constant factor) to deal with the transformations than with the dynamics of the Petri system. In addition, even though the treatment of the permutation symmetry type is complex, it is better than trying out all the possibilities and maybe even significantly better than the ordinary construction of a reachability graph.

Even if no time were saved in the symmetry method, the profit in the size of the reachability graph can considerably reduce the time spent in the further processing of the graph, in particular, processing the queries about the properties of the system.

Many theoretically interesting protocols are fully symmetrical. In practical cases, however, pure symmetries may not occur often. That diminishes the value of the theoretical machinery, though non-sound symmetries are not totally useless.

3 Parameterised Reachability Analysis

3.1 Introduction

The symmetry method of Chapter 2 yields good results and it is theoretically solid. However, it presumes that the analyst can reveal the symmetries by his knowledge, and besides, it has essentially been defined for two symmetry types only. Furthermore, even though the markings and transition instances represent equivalence classes, the method treats them at the low level as in the ordinary reachability analysis. These facts can be seen as flaws that one should get rid of. In that spirit, Markus Lindqvist has presented the *parameterised reachability analysis method* of high-level systems [12, 13].

Figures 3 to 7 illustrate the principles behind the parameterising method. Figure 3 presents a simple high-level system model of n independent processes (suppose $n \geq 2$). Transition t_1 has several possible enabled instances. Instead of firing any of them, the local variable x is bound to a so-called parameter $\hat{v}$. The parameterised marking of Figure 4 stands for the n possible follower markings; note the subtractional expression in the marking of p_1. The parameter $\hat{v}$ may assume any possible "consistent" value. Thus it represents some integer between zero and $n-1$, since other values cannot be subtracted from $\langle 0 \rangle + \cdots + \langle n-1 \rangle$.

If t_1 fires again, a new unique parameter $\hat{w}$ must be chosen (Figure 5). For the sake of consistency, $\hat{v}$ and $\hat{w}$ must assume different values. Continuing in the same manner, a reachability graph with $\binom{n+2}{2} = O(n^2)$ vertices is obtained. (The result would be the same if the high-level system were first transformed to a place/transition system by removing the identities of the tokens.) This can be contrasted to the normal reachability graph generation of the high-level system, which would produce 3^n vertices.

As can be expected, the parameterised reachability graph generation is not quite so simple. Transition t_2 may fire at the marking of Figure 5 with x assigned to $\hat{v}$ or $\hat{w}$. To represent both these instances a new parameter $\hat{x}$ is introduced, so the firing produces the marking of Figure 6. But that, in turn, can be reduced to a simpler equivalent form, which is seen in Figure 7.

The parameterised reachability analysis is discussed more thoroughly in the following sections. Instead of adopting the rather cumbersome formalism of Lindqvist's, a semiformal description is given.

80

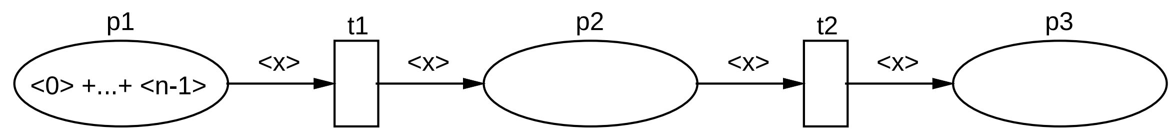


Figure 3: A high-level system model at the initial marking.

80

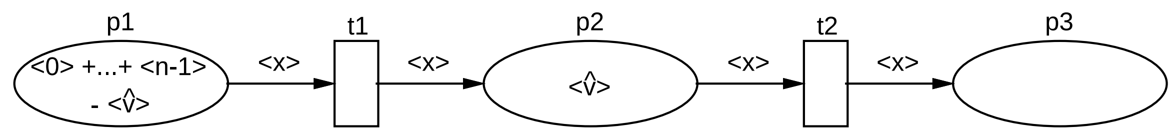


Figure 4: After the parameterised firing $t_1 : \{x \mapsto \hat{v}\}$.

80

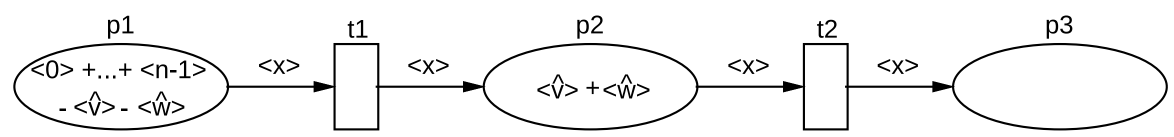


Figure 5: After the parameterised firing $t_1 : \{x \mapsto \hat{w}\}$.

80

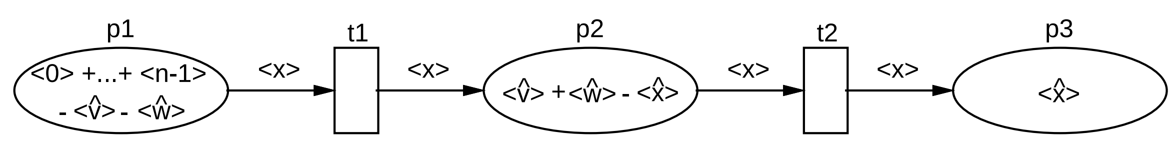


Figure 6: After the parameterised firing $t_2 : \{x \mapsto \hat{x}\}$.

80

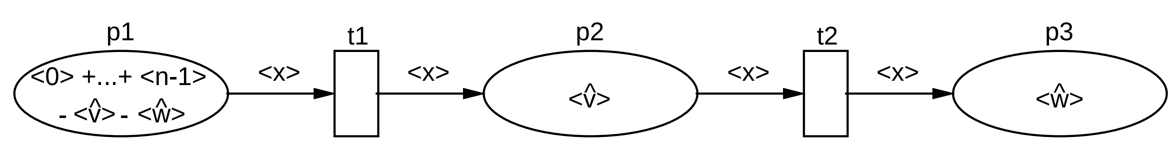


Figure 7: The same marking in the reduced form.

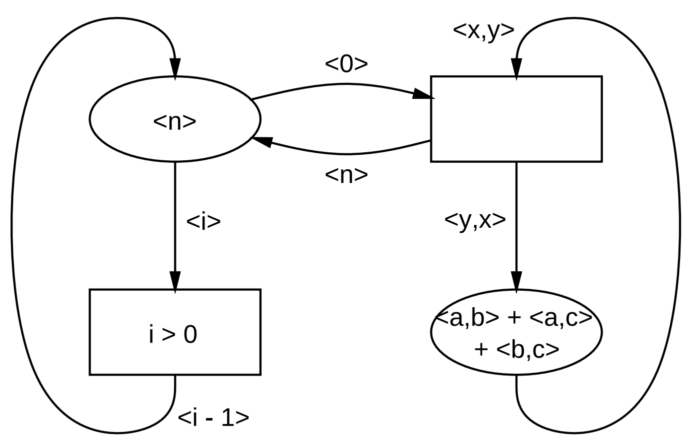


Figure 8: A PrT system.

3.2 Predicate/Transition Systems

A predicate/transition system, or PrT system for short, is a quintuple $\langle S, T, G, A, M_0\rangle$ where S is the set of places and T is the set of transitions. If t is a transition, $G(t)$ is a logical formula, the selector of t. If t is a transition and s is a place, $A(s,t)$ and $A(t,s)$ are symbolic sums, the arc expressions, representing the flow from s to t and conversely. The initial marking M_0 (and any other marking) is a function that assigns each place a symbolic sum. A sample PrT system is seen in Figure 8. Note that the empty symbolic sums (0) in a marking and arc expressions as well as the tautology ($\top$) in a selector are usually omitted in the graphical presentation of a PrT system.

Symbolic sums (eg, $\langle a,b\rangle + \langle b,x\rangle$) and formulas (eg, $x = a \vee \forall y(P(y) \Rightarrow x \neq a)$) are expressions. An expression may contain variables, which are usually denoted by letters x, y etc. If it has variables (outside the scope of quantifiers $\forall$ and $\exists$), it is called open and otherwise closed. Any closed expression e has an interpretation $\mathcal{R}(e)$, which is determined by the interpretation of its subexpressions: terms, operations, predicates and constants. Thus $\mathcal{R}(\langle a\rangle + \langle b\rangle + \langle c\rangle) = \{\mathcal{R}(a), \mathcal{R}(b), \mathcal{R}(c)\}$ and $\mathcal{R}(7 < 8+2) = \mathit{true}$ under the usual interpretation of 7, 8, 2, $<$ and $+$. Without loss of expressive power, the terms in the symbolic sums are here limited to be either constants or variables, so $\langle x+1\rangle$, for instance, is not considered a proper symbolic sum.

As symbolic sums are interpreted as sets (according to the so-called strict interpretation), expressions like $\langle a\rangle + \langle a\rangle$ or $\langle n\rangle + \langle 1\rangle$ with $\mathcal{R}(n) = 1$ have no interpretation. In other words, the '+' in the symbolic sums does not stand for union but rather a kind of addition.

Let α be a mapping that assigns values to every variable in an expression e, ie, α is a substitution. Then $e : \alpha$ is the corresponding closed instance of e. All the variables in $G(t)$, $A(s,t)$ and $A(t,s)$ are local to transition t. If α substitutes a value for each variable in $G(t)$, $A(s,t)$ and $A(t,s)$ ($\forall s \in S$), then $t : \alpha$ is called a transition instance. Transition instance $t : \alpha$ is called feasible if $\mathcal{R}(G(t) : \alpha) = \mathit{true}$ and the closed arc expressions $A(s,t)$ and $A(t,s)$ all have an interpretation, ie, they represent sets. Only the feasible instances are interesting, since only they "are there", only they have a possibility to fire.

A feasible transition instance $t : \alpha$ is enabled at marking M, denoted by $M[t : \alpha\rangle$, if

$$\forall s \in S\, [\mathcal{R}(A(s,t) : \alpha) \subseteq \mathcal{R}(M(s)) \wedge \mathcal{R}(A(t,s) : \alpha) \cap \mathcal{R}(M(s)) = \emptyset].$$

A transition instance $t : \alpha$ enabled at M can fire to marking M', denoted by $M[t : \alpha\rangle M'$, if

$$\forall s \in S\, \mathcal{R}(M'(s)) = \mathcal{R}(M(s)) \setminus \mathcal{R}(A(s,t) : \alpha) \cup \mathcal{R}(A(t,s) : \alpha).$$

The reachability graph of a PrT system can now be constructed. Let V be the set of vertices and E the set of edges. The initial marking $M_0 \in V$. If $M \in V$ and $M[t : \alpha\rangle M'$, then $M' \in V$ and $\langle M, t : \alpha, M'\rangle \in E$. There are no other members in V and E.

3.3 Parameterised Reachability Graph Generation

In the definition of the dynamics of PrT systems it is immediately seen that it is not transitions that fire but transition instances. As a result, the PrT systems, although being a high-level net formalism, have a low-level semantics. It is exactly as burdensome to construct the reachability graph of a PrT system model as that of the equivalent elementary net system. In the parameterised reachability graph generation all instances of a transition can in principle be represented by the transition itself.

To attain the goal, parameters are introduced. While the markings in the ordinary reachability graphs consist of constants, the markings of the parameterised reachability graphs have constants and parameters, which stand for "any constant". Parameterised entities are denoted by hatted letters (eg, $\hat{v}$ is a parameter and $\widehat{M}$ is a parameterised marking). Thus the parameterised marking

$$\widehat{M} = \left\{ \begin{array}{l} s_1 \mapsto \langle a, \hat{v}\rangle + \langle \hat{v}, \hat{w}\rangle \\ s_2 \mapsto \langle \hat{v}\rangle + \langle \hat{w}\rangle \end{array} \right\}$$

may represent marking

$$M_1 = \left\{ \begin{array}{l} s_1 \mapsto \langle a, b\rangle + \langle b, c\rangle \\ s_2 \mapsto \langle b\rangle + \langle c\rangle \end{array} \right\}$$

or

$$M_2 = \left\{ \begin{array}{l} s_1 \mapsto \langle a, c\rangle + \langle c, a\rangle \\ s_2 \mapsto \langle c\rangle + \langle a\rangle \end{array} \right\}$$

but not

$$M_3 = \left\{ \begin{array}{l} s_1 \mapsto \langle a, b\rangle + \langle b, b\rangle \\ s_2 \mapsto \langle b\rangle + \langle b\rangle \end{array} \right\}$$

which is no marking at all since $\langle b\rangle + \langle b\rangle$ cannot be interpreted as a set. In addition, parameterised markings may contain "negative" tuples: $\langle a\rangle + \langle b\rangle - \langle \hat{v}\rangle$ stands either for $\langle a\rangle$ or $\langle b\rangle$.

When a transition t "is about to fire" at a parameterised marking $\widehat{M}$, each of the variables of t is assigned a unique parameter not occurring in the marking. Then by a kind of unification one tries to replace as many parameters as possible by constants or parameters that occur in the marking. Let the resulting unification be $\hat{\alpha}$. Then t is α-enabled at $\widehat{M}$, if there is a marking M and a substitution α such that

1. $\widehat{M}$ can represent M,
2. $\hat{\alpha}$ can represent α and
3. $M[t : \alpha\rangle$.

This is denoted by $\widehat{M}[t : \hat{\alpha}\rangle$.

It may seem that $t : \hat{\alpha}$ is another form of a transition instance but it is not so if one also requires that $t : \hat{\alpha}$ represents every enabled transition instance. A parameterised marking has then at most one follower edge for each transition while an ordinary marking can have several follower instances of the same transition.

If a transition t is $\hat{\alpha}$-enabled at $\widehat{M}$, it can fire to the parameterised marking $\widehat{M}'$ that is obtained from $\widehat{M}$ in the following way:

1. For each $s \in S$, set $\widehat{M}'(s) = \widehat{M}(s) - A(s,t) : \hat{\alpha} + A(t,s) : \hat{\alpha}$.
2. Reduce $\widehat{M}'$ as much as possible so that it represents the same set of markings (eg, $\langle a\rangle - \langle \hat{v}\rangle$ may be reduced to 0).

This denoted by $\widehat{M}[t : \hat{\alpha}\rangle \widehat{M}'$.

Let $\mathcal{V}(\widehat{M})$ be the set of markings $\widehat{M}$ can represent. The parameterised reachability graph is a pair $\langle \widehat{V}, \widehat{E}\rangle$. The set containing the initial marking $\{M_0\} \in \widehat{V}$. If $\mathcal{V}(\widehat{M}) \in \widehat{V}$ and $\widehat{M}[t : \hat{\alpha}\rangle \widehat{M}'$, then there is a set $\mathcal{M}$ of markings such that

$\mathcal{M} \in \widehat{V}$, $\mathcal{V}(\widehat{M}') \subseteq \mathcal{M}$ and $\langle \mathcal{V}(\widehat{M}), t : \hat{\alpha}, \mathcal{M} \rangle \in \widehat{E}$. There are no other members in $\widehat{V}$ and $\widehat{E}$.

The $\mathcal{M}$ above refers to how the reachability graph generation algorithm functions: if paths are found from M_0 to parameterised markings $\widehat{M}_1$ and $\widehat{M}_2$ such that $\mathcal{V}(\widehat{M}_1) \subseteq \mathcal{V}(\widehat{M}_2)$, then $\mathcal{V}(\widehat{M}_2)$ is the $\mathcal{M}$ corresponding to $\mathcal{V}(\widehat{M}_1)$ and the paths leading out of $\mathcal{V}(\widehat{M}_1)$ are not examined.

3.4 Discussion

Figure 9 presents the PrT system model and parameterised reachability graph of two dining philosophers. At the initial marking, only transition t_L may fire. As x cannot be bound to either of the values 1 and 2, it is assigned a parameter $\hat{v}$ so the parameterised marking $\widehat{M}_1$ is obtained. Now both t_L and t_R are enabled. Firing of t_R with y mapped to a new parameter leads to $\widehat{M}_2$. But by noticing that place F must be empty, $\widehat{M}_2$ can be reduced to $\widehat{M}_3$. At this marking, only r_L can fire producing marking $\widehat{M}_4$. Now t_L is not enabled, since x should bound to $\hat{v}$ because of place F but $\hat{v}$ is not there in place P. Firing transition t_R yields $\widehat{M}_5$, which can be reduced to M_0, since the domain is $D = \{1, 2\}$. If it is t_L that fires at $\widehat{M}_1$, $\widehat{M}_6$ is obtained. It can be reduced to M_7, which is a deadlock.

By now, several questions have arisen. The generation process did not seem at all obvious. Are there precisely formulated algorithms for accomplishing the different steps of the reachability graph generation? The parameterised analysis found the only deadlock of the system. However, the result seemed to appear rather by accident or by the size two of the domain than by strict correspondence to the intuitive operation of the system. More specifically, the process seems to lose the information about the origin of the parameters. Will the method work with any number of philosophers? Will it work on any system? What system properties are preserved? These questions are dealt with in the sequel.

The parameterised reachability analysis of the three-philosopher problem is considerably more complex than that of two philosophers. Nevertheless, the method will find the deadlock regardless of the number of the philosophers. The strict interpretation of PrT systems with the introduction of the negative tuples makes the method work well on many "usual" problems, but the loss of information may cause failure in detecting deadlocks as in Figure 10 (which is a modified version of Fig 5.1. of [12]). After T has fired, the system is in a deadlock since B has neither $\langle 1 \rangle$ nor $\langle 2 \rangle$. The parameterised firing, however, puts $\langle \hat{v} \rangle$ into C and forgets that it cannot be $\langle 3 \rangle$. Thus U can fire with $x \mapsto 3$, so the system looks live.

53

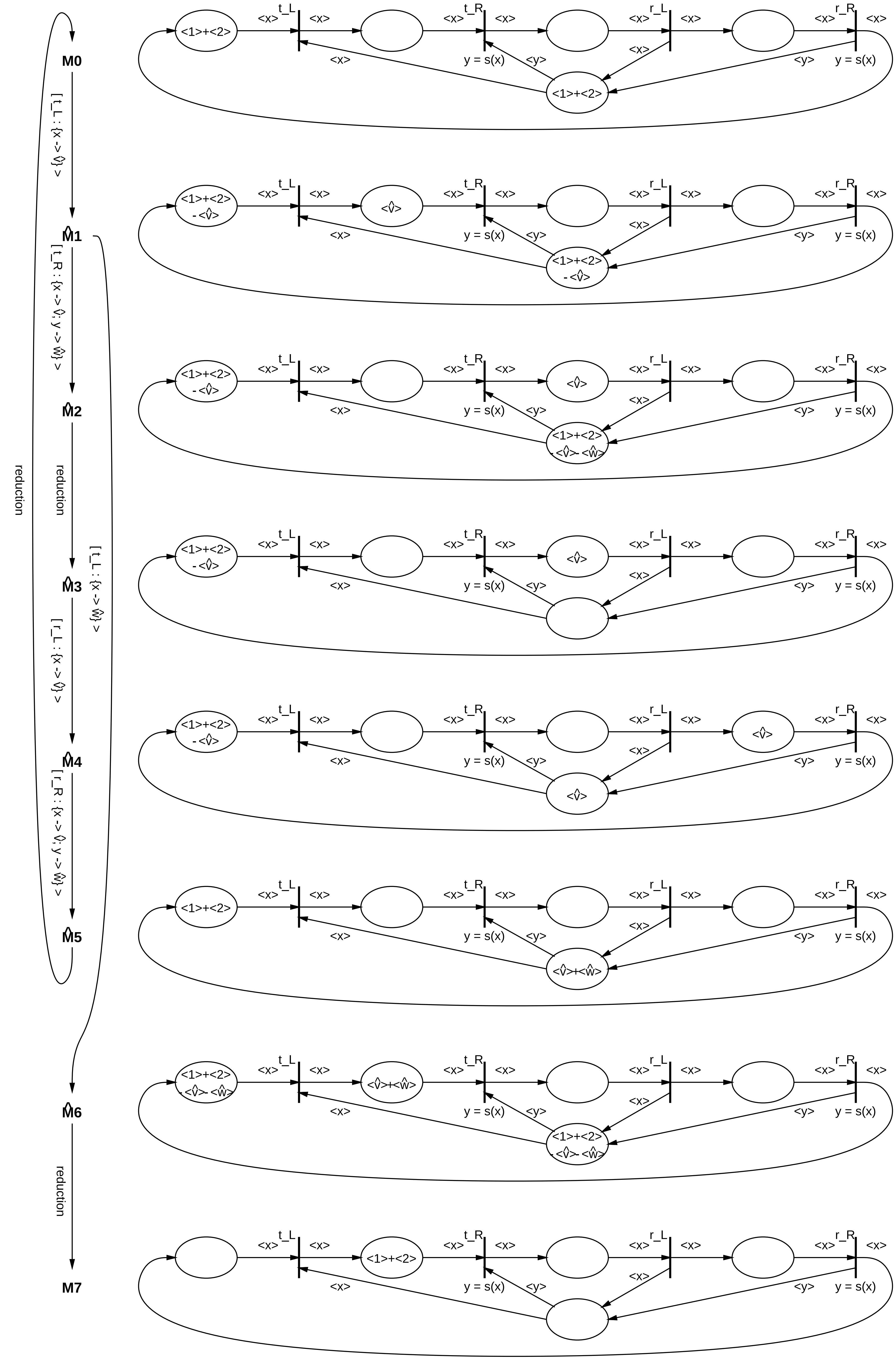


Figure 9: The parameterised reachability graph of two dining philosophers.

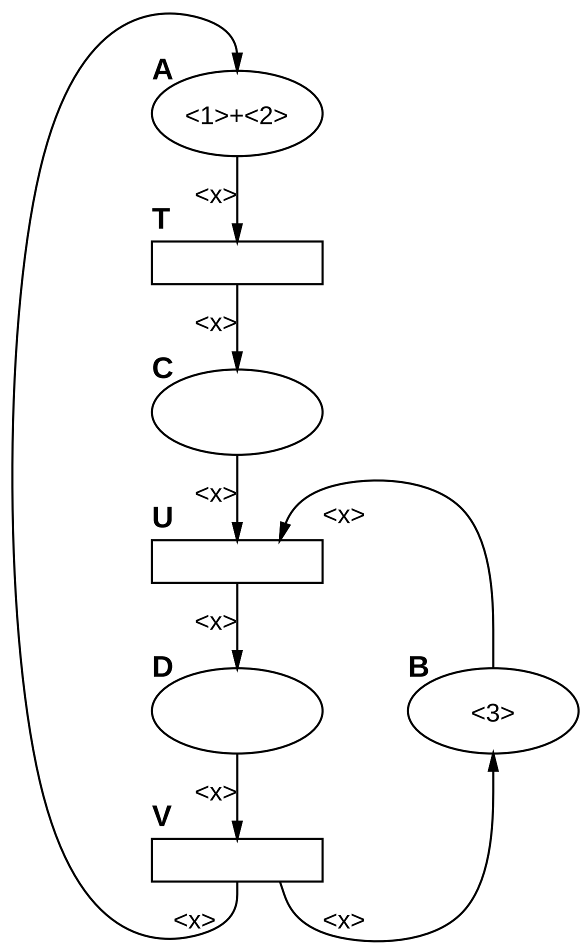


Figure 10: Parameterising loses a deadlock.

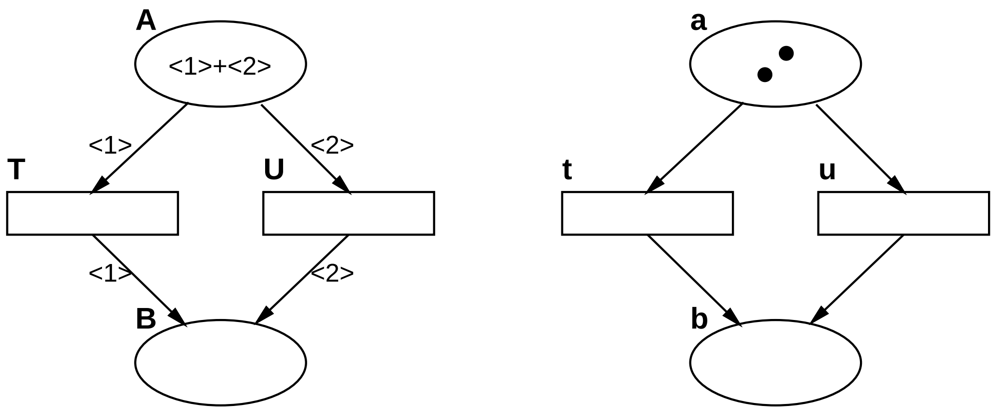


Figure 11: A "pathological" PrT system and the corresponding P/T system.

Many algorithms are needed in the various phases of firing a transition.

1. The tuples in the arc expressions must be matched with the tuples of the marking by assigning constants and parameters to the variables in the arc expressions. Since there may be many possibilities to couple the tuples of the arc expressions with those in the marking, the best algorithm is little better than trying out every possible combination.

2. After matching it is still possible that the transition is not enabled. That can be checked by an again combinatorially hard algorithm.

3. After an enabled transition is fired, the resulting marking should be reduced, though it is not necessary to attain the optimal reduction. There is no efficient general algorithm for the job.

4. The reduced marking must be compared to the already generated markings to see whether it is included in any of them. This is similar to the matching problem.

(Lindqvist does not give the algorithms in an explicit form but he defines partial algorithms like matching markings of places instead of those of the whole system.)

Thus a tremendous effort is needed when the parameterising method is applied. What then are the advantages of using it? Lindqvist has obtained three results, which are weak indeed.

1. If the domain D is finite, then the parameterised reachability graph of any system is finite.

2. If the PrT system under inspection is transformed to a P/T system by removing the identities of the tuples, reducing the arc expressions to their cardinalities and removing the selector formulas, then, under strong additional assumptions, the reachability graph of the P/T system is not smaller than the parameterised graph of the original PrT system. For a counter-example (which does not satisfy the assumptions), see Figure 11

3. If there is a path
$$M_1[t_1 : \alpha_1\rangle M_2 \cdots [t_{n-1} : \alpha_{n-1}\rangle M_n$$
in the ordinary reachability graph of any PrT system, then there is a path
$$\widehat{M}_1[t_1 : \hat{\alpha}_1\rangle \widehat{M}_2 \cdots [t_{n-1} : \hat{\alpha}_{n-1}\rangle \widehat{M}_n$$
with $M_i \in \mathcal{V}(\widehat{M}_i)$ and $t_i : \alpha_i \in \mathcal{V}(t_i : \hat{\alpha}_i)$ in the parameterised reachability graph of the system.

The last statement expresses that the parameterising method is complete: it contains the whole of the system's operation as a subgraph. But since the opposite is not true (the method often generates nonexistent system states), it does not give any picture of how good the method is. For the finite graph containing every constructible marking and transition instance would trivially fulfil the same requirement.

3.5 Conclusion

A high-level reachability analysis by a folded or symbolic reachability graph generation is the goal of Lindqvist's parameterising method. However, the involved definitions make the method difficult to automate. The algorithms necessary in the generation are both incomplete and extremely laborious. Finally, the relationship between the ordinary reachability graph and the parameterised one has not yet been characterised nearly at all.

In fact, it seems that most cases where the method works well take advantage of the rather arbitrary requirement of strictness (which forces two tuples to assume different values). The concepts of parameters and negative tuples can cover but a few of the extensive set of possible relationships between individuals.

Thus, unfortunately, at this stage the parameterising method is far from a theoretically clear, practical means for reachability analysis. The least requirement for further research is to show what system properties can be analysed and under which restrictions.

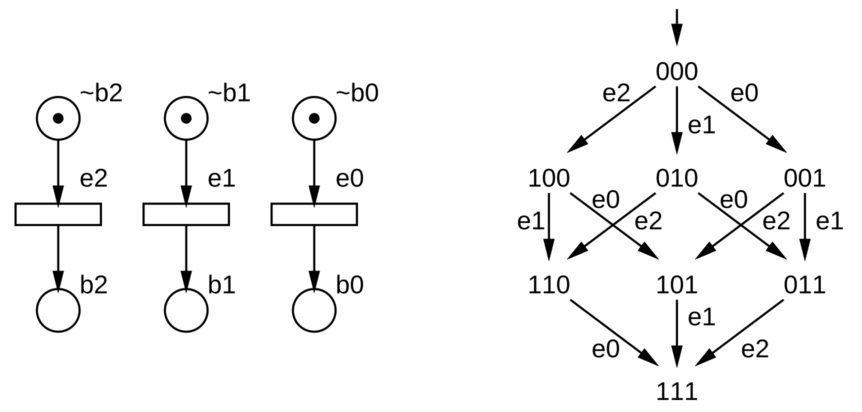


Figure 12: A Petri system model and its reachability graph.

4 Stubborn Set Method

4.1 Introduction

Figure 12 presents a low level Petri system model with its corresponding reachability graph. The operation defined by the model is simple and easy to understand. The reachability graph, instead, is fairly complex (even though the concurrent occurrences of the events have been left out). It contains eight states and six different paths from the initial state to the final state.

Lattices like this are very common in practical system models and they are to a great extent responsible for the big size of a reachability graph. On the other hand, there are important reachability problems (eg, finding deadlocks) from whose point of view the path variants are equivalent. Indeed, methods have been developed to make reachability analysis more efficient by partly recognising and eliminating event sequences which are unnecessary for solving the task.

In the *priority method*, the analyst beforehand seeks to recognise the structures which cause problems and gives the events a partial priority order. If events with different priorities are concurrently enabled, only those with the highest priority are allowed to occur. The events of the example of figure 12 may be given a priority order according to which, say, $e_2 \prec e_1 \prec e_0$. Then in the initial state only e_2 may occur, and the only event sequence goes along the leftmost route of the lattice in the figure.

A restricted form of the priority method is the *virtual coarsening of atomic actions* [15] in which events local to a process are recognised and made to occur immediately when they have become enabled. For instance, in figure 13

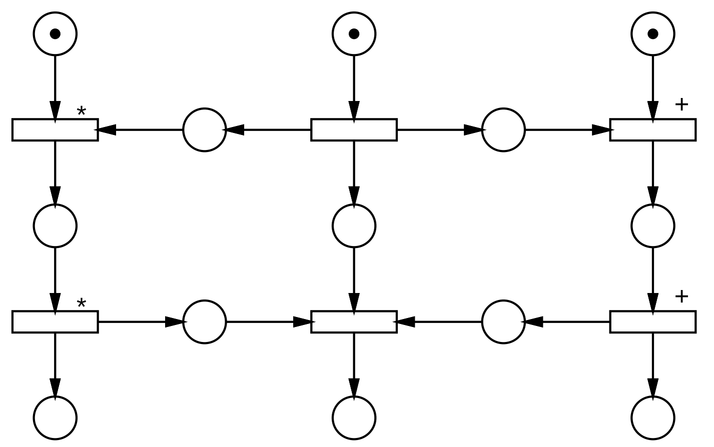


Figure 13: Events marked with a star (plus) may occur atomically.

there can be thought to be three processes. The process in the middle sends a message to the others, which receive it and eventually thereafter send a reply message. The model might be faithful to the real system but the fact that the events marked with a star (or plus) are separate makes the analysis more complex than necessary. One can therefore require that the events of reply occur before any other event after they have become enabled.

Antti Valmari has presented the *stubborn set method* [18, 19, 20, 21, 22, 23], which tries to solve the "equivalent path problem" in a more thorough way than the priority methods. The stubborn set method is a sort of dynamic priority method. Instead of priorising the events solely by examining the structure of the model in its initial state, the priorised events are computed separately in each state.

One can loosely say that a stubborn set consists of events whose occurrence cannot be affected by any other events. Figure 12 has several stubborn sets. For instance $\{e_1\}$ is stubborn since its member e_1 will occur regardless of the occurrence of the other events. Also, the set of all events is trivially stubborn. In the example the stubborn set method seems to be the same as the priority method but a case will be presented later where the stubborn sets reveal their superiority.

Since stubborn sets are searched separately in each state, the computation ought to be fast. The computation of the minimal stubborn set typically takes a quadratic time with respect to the number of events, which must be considered infeasible. Valmari has, however, developed an algorithm that in linear time finds an often good stubborn set. The algorithm has variations for examining different system properties [22, 23].

The predecessor of Antti Valmari's work is William Overman. In his doctoral thesis he presented two methods [14] that resemble the stubborn set method but are more restricted.

4.2 Definition of Stubborn Sets

The principle of stubborn sets can be defined for any modelling formalism with events that depend on the local state. Valmari presents his definition to variable/transition systems and different kinds of Petri systems [22]. In the following, a variant of place/transition systems is presented.

4.2.1 Place/Transition Systems

Definition 4.1 *A* place/transition system *(with infinite capacities), or a P/T system for short, is a quadruple $\langle S, T, W, M_0 \rangle$, where S is the set of places, T is the set of transitions ($S \cap T = \emptyset$), $W : S \times T \cup T \times S \to N$ is the arc weight function and $M_0 : S \to N$ is the initial marking.*

If $x \in S \cup T$, then one denotes $\bullet x = \{y \,|\, W(y,x) > 0\}$ and $x \bullet = \{y \,|\, W(x,y) > 0\}$.

In the sequel, one always refers to the P/T system $\langle S, T, W, M_0 \rangle$.

P/T systems are given an interleaving semantics.

Definition 4.2 *A* marking *is a function $M : S \to N$. Transition $t \in T$* is enabled at marking *M (denoted $M[t\rangle$) if $\forall s \in \bullet t\; M(s) \geq W(s,t)$. Transition $t \in T$ enabled at marking M* can fire to marking *M' (denoted $M[t\rangle M'$) if $\forall s \in S\, M'(s) = M(s) - W(s,t) + W(t,s)$. Marking M* is dead *if $\forall t \in T \neg M[t\rangle$.*

Definition 4.3 *The* reachability graph *of a P/T system is a pair $\langle V, E \rangle$, where V is a set of markings (vertices) and $E \subseteq V \times T \times V$ (edges) and the following holds:*

1. *$M_0 \in V$.*
2. *If $M \in V$ and $M[t\rangle M'$, then $M' \in V$ and $\langle M, t, M' \rangle \in E$.*
3. *V and E have no other elements.*

Definition 4.4 *Let $\sigma = \langle t_1, t_2, \ldots, t_n \rangle \in T^*$ and let M be a marking. One denotes $M[\sigma\rangle$ if there exist markings $M_1, M_2, \ldots, M_{n-1}$ such that*

$$M[t_1\rangle M_1[t_2\rangle M_2 \cdots [t_{n-1}\rangle M_{n-1}[t_n\rangle.$$

Correspondingly, one denotes $M[\sigma\rangle M'$ *if there exist markings* $M_1, M_2, \ldots, M_{n-1}$ *and* M' *such that*

$$M[t_1\rangle M_1[t_2\rangle M_2 \cdots [t_{n-1}\rangle M_{n-1}[t_n\rangle M'.$$

The basic formalism has now been defined. An introduction to the stubborn sets follows.

4.2.2 The Idea behind Stubborn Sets

Suppose a P/T system can be divided into a "black box" and its environment. The environment may or may not fire transitions before anything happens in the black box. The partition has, however, been accomplished in such a manner that the next action (firing of a transition) of the black box is fully independent of the operation of the environment and, moreover, the action of the black box does not affect the operation of the environment. In the traditional reachability analysis, a lattice of similar paths would now appear, as described in the introduction. A situation like this can be condensed to the following principles.

Principle 1:

$$\text{If } \neg M[t\rangle \text{ and } M[\sigma\rangle, \text{ then } \neg M[\sigma t\rangle,$$

where M is the marking where the system was partitioned, t is any transition of the black box, and σ is any transition sequence of the environment. The principle states that if the black box has disabled transitions at M, they cannot become enabled as a result of the operation of the environment. The first principle is unnecessary if only enabled transitions are taken in the black box.

Principle 2:

$$\text{If } M[t\rangle \text{ and } M[\sigma\rangle, \text{ then } M[\sigma t\rangle \text{ and } M[t\sigma\rangle.$$

The second principle expresses the fact that it does not make any difference how the transition of the black box and the operation of the environment are interleaved. Note that according to the semantics of P/T systems the order of firing does not affect the resulting marking.

One of the most important reachability problems is finding the dead markings because a dead marking usually means a severe design or modelling error or is a result of a terminating computation. The two principles presented above give permission to choose one of the enabled transitions of the black box to be fired first without investigating the other interleavings. Namely, let there be an enabled transition in the black box and also a path leading to a dead marking. Then the path must contain at least one of the enabled transitions of the black box. For if it were not so, according to Principle 1, the path would not contain any of the disabled transitions of the black box and hence, according to Principle 2, the enabled transitions of the black box would still be enabled in the end of the path. But in that case, according to Principle 2, the same marking can be achieved by transferring a transition of the black box to the beginning of the sequence. Thus any dead marking will be found by firing only transitions of the black box whenever it is possible.

On the basis of the presented principles it would be possible to define a feasible stubborn set method, but a little looser conditions suffice for finding the dead markings, namely, the following two principles.

Principle 1:*

$$\text{If } M[\sigma t\rangle, \text{ then } M[t\sigma\rangle,$$

where M, again, is the marking where the system was partitioned, t is a transition of the black box (stubborn set), and σ is a transition sequence of the environment.

Principle 2:*

$$\text{If } M[\sigma\rangle, \text{ then } M[\sigma\tau\rangle,$$

where τ is a fixed transition in the stubborn set. So, a stubborn set is never empty. The environment cannot prevent τ from firing.

The dead markings are found again. Let $v \in T^*$ and $M[v\rangle M'$ and let M' be a dead marking. If v consists of transitions of the environment only, according to Principle 2*, $M'[\tau\rangle$. But since M' is dead, v must contain a transition of the stubborn set, ie, it has the form $\sigma t v'$. Now Principle 1* implies that $M[t\sigma v'\rangle M'$, which was to be proved. Note that the new route is exactly as long as the original one.

According to the new principles, some transitions of the stubborn set may become disabled as a consequence of the actions of the environment. As always only stubborn transitions are fired, those transition sequences seem to be left uninvestigated. This is, however, not true since there is always at least one "well-behaved" stubborn transition τ. After firing τ and computing a new stubborn set, the tricky path is still possible and will eventually be examined. (See also the row (2c) of Definition 4.5 below and Figure 14.)

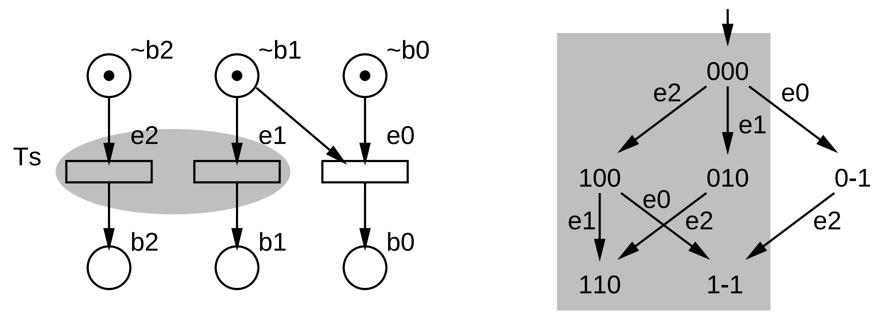


Figure 14: A stubborn set and the reduced part of the reachability graph.

4.2.3 Stubborn Sets

The principles presented in the preceding do not form a method which helps the reachability analysis because evaluating them itself requires at least partial reachability analysis. A practical method requires investigating only the structure of the P/T system with the marking, or in other words, the method is structural. Valmari defines for P/T systems a structural criterion for so-called semistubbornness, which fulfils Principle 1*. The definition is rather complex, so it is reasonable to present a couple of abbreviations: one denotes $M(s)[t\rangle$ if $M(s) \geq W(s,t)$; and $\Delta(t,s) = W(t,s) - W(s,t)$.

Definition 4.5 *A set of transitions $T_s \subseteq T$ is* semistubborn at marking M *if*

$\forall t \in T_s$ (

(1) $\exists s \in \bullet t\, [\neg M(s)[t\rangle \wedge \forall t' \in T \setminus T_s\, (M(s)[t'\rangle \Rightarrow \Delta(t',s) \leq 0)]$

$\vee$

(2) $\forall s \in \bullet t\, \{M(s)[t\rangle \wedge$

(2a) $[(\Delta(t,s) \geq 0 \wedge \forall t' \in T \setminus T_s\, \Delta(t',s) \geq 0)$

$\vee$

(2b) $\forall t' \in T \setminus T_s\, (\Delta(t',s) \geq 0 \wedge W(s,t') \leq M(s) + \Delta(t,s))$

$\vee$

(2c) $\forall t' \in T \setminus T_s\, (M(s)[t'\rangle \Rightarrow \Delta(t',s) \leq 0 \wedge W(t',s) \leq W(t,s))\})$.

Row (1) in the definition fulfils Principle 1: the environment $T \setminus T_s$ must not enable a transition not enabled at marking M. Thus Principle 1* is not violated, either. The possibility of including disabled transitions in a semistubborn set significantly improves the efficacy of the method.

Rows (2) to (2c) deal with what kinds of relationships are allowed between an enabled stubborn transition t and its input places $\bullet t$. Row (2a) requires that t must not prevent the transitions of the environment from firing and the environment, in turn, must not prevent t from firing. The goal is achieved by forbidding t and t' to decrease the marking of the communication place. The same goal can be attained by another condition: row (2b) utilises the fact that t is fired only once, so its blocking effect can be calculated exactly. Of the similar rows (2a) and (2b), the less restricting is chosen in every situation.

Row (2c) does not fulfil Principle 2. This simple structural backdoor lets in a little more difficultly behaving transitions into the semistubborn set. Figure 14 presents a semistubborn set in which e_1 is included by virtue of row (2c).

Theorem 4.6 *Any semistubborn set fulfils Principle 1*.*

A straightforward proof of the theorem has been presented in [18].

As can be expected, the structural semistubbornness is not the most unrestricted definition fulfilling Principle 1*. It is, however, difficult to extend it in a natural way. Valmari has presented several versions of the definition varying a little its complexity and generality.

The stubborn set, which fulfils Principles 1* and 2*, is defined next.

Definition 4.7 *A set of transitions $T_s \subseteq T$ is* stubborn at marking M *if it is semistubborn at marking M and*

$$\exists \tau \in T_s \, \forall s \in \bullet\tau \, (M(s)[\tau\rangle \;\wedge\; \forall t' \in T \setminus T_s \, \Delta(t', s) \geq 0).$$

The semistubborn set of Figure 14 is obviously stubborn, too.

Theorem 4.8 *Any stubborn set fulfils Principles 1* and 2*.*

The proof is obvious: the environment cannot prevent τ from firing.

The definition of the stubborn set thus provides a structural reduction method of a reachability graph. Its usability is dealt with in the following sections.

4.2.4 Example: Dining Philosophers

Valmari presents the problem of dining philosophers as an application example of the stubborn set method [18]. A high level Petri system model of the problem is in Figure 15. Valmari shows that in the reachability graph of the system there are $3^n - 1$ vertices and $n \cdot (2 \cdot 3^{n-1} - 1)$ edges.

If one constructs a reduced reachability graph by choosing suitable stubborn sets, there will be left only $3n^2 - 3n + 2$ vertices and $4n^2 - 3n$ edges. This kind

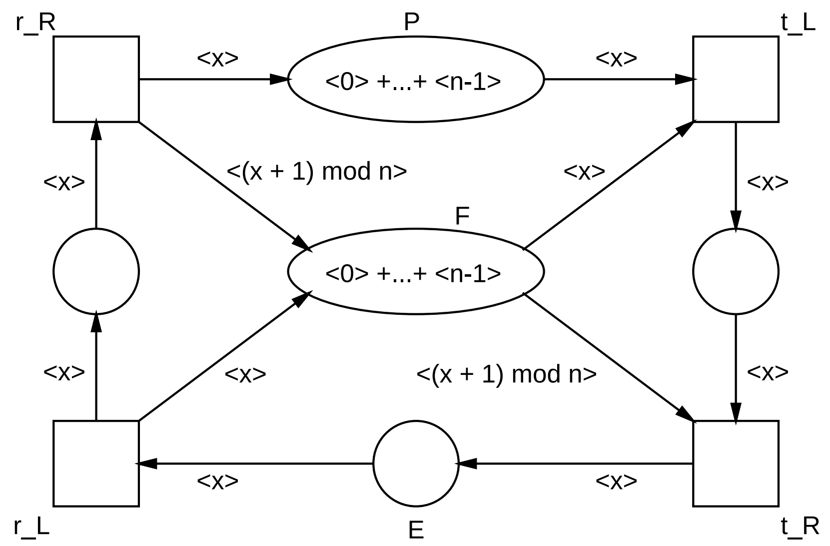


Figure 15: Dining philosophers.

of suitable choice can be made by taking at each marking as the stubborn set the first possible from the following list:

$$\{r_L[0]\}, \ldots, \{r_L[n-1]\}, \{r_R[0]\}, \ldots, \{r_R[n-1]\},$$
$$\{t_L[0], t_R[n-1]\}, \{t_L[1], t_R[0]\}, \ldots, \{t_L[n-1], t_R[n-2]\},$$
$$T.$$

The proof is presented in [18].

If the symmetry of the problem is further utilised by assuming that the transition instance $t_L[0]$ will fire at the initial marking, the number of vertices becomes $3n-1$ and that of edges $4n-3$.

The example shows thus that the stubborn set method may even drop a state-space exponential with respect to the size of the net to quadratic. If other known methods are used in addition, still better results may be achieved.

Thus, in applying the method there is much to win but, on the other hand, it would seem to require plenty of computation at every marking. In the following the computation of stubborn sets and its efficiency is considered.

4.3 Computation of Stubborn Sets

The definition of a stubborn set implies that at all living markings there is at least one stubborn set T (the set of all transitions). It would usually be best to find a stubborn set with a minimal number of enabled transitions. (Although it may not always lead to the best possible reduction, it is difficult to define a better simple goal.) There are algorithms for that purpose, and some of them are described in the following subsections.

4.3.1 Candidate List Algorithm

In this approach the analyst presents a due list of candidates for the stubborn sets before the generation of the reachability graph. The last element of the list is T. The candidates are tried out in turn at every (live) marking until one fulfilling the definition of the stubborn set is found. This algorithm was applied in the example of the previous section.

The algorithm has two advantages. First, it is not necessary to traverse through all possible transition sets but only a considerably shorter list. Secondly, the stubbornness test is very fast because a great deal of the definition is independent of the marking and can be precomputed.

The algorithm as given by Valmari takes a time of at most $O(c\sum_{i=1}^{n}|T_i|)$, where $c = \max_{t\in T}|\bullet t|$ and the T_i are the elements of the candidate list. The algorithm is thus very quick but the candidate list given by the analyst determines the quality of the results.

4.3.2 Algorithm using Strongly Connected Components

The definition of the semistubborn set makes it possible to enumerate implications of the form "if transition t is semistubborn so must the transitions of the set $E(t)$ be". Then one semistubborn transition generates a semistubborn closure which may be a proper subset of T.

Let a marking M and a transition t be given. The set $E(t)$ is constructed on the basis of Definition 4.5. If $\neg M[t\rangle$, then row (1) of the definition holds:

$$\exists s \in \bullet t\,[\neg M(s)[t\rangle \;\wedge\; \forall t' \in T \setminus T_s\,(M(s)[t'\rangle \Rightarrow \Delta(t', s) \leq 0)].$$

Given a place $s \in \bullet t$ such that $\neg M(s)[t\rangle$, any semistubborn set T_s must have the transitions of the set

$$E_1(t, s) = \{t' \in T \mid M(s)[t'\rangle \;\wedge\; \Delta(t', s) > 0\}.$$

The place s is chosen nondeterministically and $E(t) = E_1(t, s)$.

If $M[t\rangle$, then every $s \in \bullet t$ requires a set of semistubborn transitions. Rows (2a) and (2b) are very much alike:

$$\begin{gathered}(\Delta(t,s) \geq 0 \ \wedge \ \forall t' \in T \setminus T_s \ \Delta(t',s) \geq 0) \\ \vee \\ \forall t' \in T \setminus T_s \ (\Delta(t',s) \geq 0 \ \wedge \ W(s,t') \leq M(s) + \Delta(t,s)).\end{gathered}$$

If a given s fulfils the condition $\Delta(t,s) \geq 0$,

$$E_2(t,s) = \{t' \in T \mid \Delta(t',s) < 0\}$$

is chosen. Otherwise its superset

$$E_2(t,s) = \{t' \in T \mid \Delta(t',s) < 0 \ \vee \ W(s,t') > M(s) + \Delta(t,s)\}.$$

is chosen. Now

$$E(t) = \bigcup_{s \in \bullet t} E_2(t,s).$$

The algorithm does not utilise row (2c) because one could then lose the stubborn transition required by Definition 4.7.

Now it is possible to define the graph $\langle T, R\rangle$ where

$$t_1 R t_2 \text{ iff } t_1 \neq t_2 \text{ and } t_2 \in E(t_1).$$

A terminal strongly connected component of the graph is $T_s \subseteq T$ for which holds

1. $T_s \neq \emptyset$,
2. $\forall t_1 \in T_s \, \forall t_2 \in T_s \; t_1 R^* t_2$, and
3. $\forall t_1 \in T_s \, \forall t_2 \in T \setminus T_s \; \neg t_1 R^* t_2$.

Then T_s is semistubborn.

Tarjan's algorithm searches the strongly connected components of a graph [17]. As soon as it finds one with an enabled transition τ, the stubborn set is chosen to be $E^*(\tau)$, ie, the set of transitions reachable by E from τ.

The time complexity of the algorithm is $O(c^2|T|)$, c being the maximal number of arcs associated to a place or a transition. Its speed is thus tolerable, even good. The nondeterminism continues to be a problem: although the algorithm is sure to find a stubborn set, one can only hope that some transitions are left out of it. At any rate, the algorithm chooses the stubborn sets of the philosopher problem in the same way as in the example and there is no nondeterminism.

4.3.3 Other Algorithms

The *deletion algorithm* uses the full definition of stubbornness. One begins with the set of all transitions and asks: "if I delete this transition, which other transitions shall I have to delete, too?" Those transitions are removed if at least one enabled transition remains. The test and deletion is performed for each transition and the result is a minimal stubborn set in the sense that it could not be further reduced. (The order of transitions determines the stubborn set and the selected one may no have the least number of enabled transitions.) Since the computation time of checking one transition is $O(c^2|T|)$, the complexity of the whole algorithm is $O(c^2|T|^2)$ [19].

The stubborn set method finds any deadlocks of a system, or more generally, all such states where the only possible stubborn set is the set of all transitions. The most serious draw-back of the method is perhaps the ignoring problem: a live transition might never be fired. This leads to a possible failure of the important fact analysis. Valmari has also defined a stricter ("strong") form of stubbornness, which fulfils Principles 1 and 2 [22]. Using Tarjan's algorithm in a new way, one can detect and eliminate the ignoring phenomenon [22].

A large class of system properties can be expressed as logical formulas. Whether the system has the studied properties can be found out by a procedure called model checking. Linear Temporal Logic (LTL) is one of those that could be used in expressing the properties. Suppose there is a set of interesting LTL formulas consisting of the $\Box$ (henceforth), $\Diamond$ (eventually) and $\mathcal{U}$ (until) operators and any subformulas local to a state; ie, the formulas do not contain the $\bigcirc$ (next) operator. The stubborn set method can now be used so that it preserves the "validity" of the formulas: a formula is true in the initial marking of the original reachability graph if and only if it is true in the initial marking of the reduced reachability graph. This is accomplished by calling the transitions that may influence on the truth of the local subformulas "visible transitions" and ensuring that

- every selected stubborn set either contains all of the visible transitions or no enabled visible transition,
- when possible, every selected stubborn set contains at least one enabled invisible transition, and
- every cyclic path contains at least one state whose stubborn set contains all visible transitions.

The depth-first algorithm given by Valmari involves backtracking, and he does not give any complexity results for it [23].

Valmari has studied the relationship between his method and the symmetry method (see Chapter 2), and he has concluded that they are complementary [22]. Both Valmari's and the symmetry method could alone drop the state-space of a sample case from exponential to quadratic but when applied together, they produced a linear state-space!

4.4 Conclusion

The problem caused to reachability analysis by equivalent interleavings is generally known but it is the theory of stubborn sets which was the first to give a full idea of it. The definition of stubborn sets being of structural character is a sufficient though not necessary condition for fulfilling the aforementioned Principles 1* and 2*.

Valmari has developed a number of efficient algorithms to solve the computational problems contained in the method. The most promising stubborn set searching algorithm contains suspicious nondeterminism. Practical experience must be obtained before a final judgment can be given to the method.

In this presentation the general clear idea of stubbornness (Principles 1 and 2, or 1* and 2*) and its structural complex implementation were separated. In his papers Valmari treats the subject only in low level detail partly because he simultaneously wants to present a suitable modelling formalism of his own (variable/transition systems) for generalising the concept of stubbornness. The stubborn set method should draw proper attention among all those who study parallel and distributed systems. The difficultly intelligible definitions along with the unvisual denotations can, however, prove an obstacle to the spreading of the method.

5 Reduction Theory

5.1 Introduction

A system modeller must continuously make decisions about which aspects of the system are to be modelled and in which detail. For instance, should two subprogrammes be modelled by two distinct transitions or could they be represented by one atomic transition? What features of the system would then be preserved?

The choices have a direct effect on how complicated the reachability analysis of the system will turn out. Usually, the more direct the correspondence between the model and the real system is, the less feasible the reachability analysis appears to be. On the other hand, the more simplifications are made, the more questionable the equivalence between the model and the system will become.

The *reduction theory* of Petri systems aims at controlled simplification of net models. The modeller can apply *reduction rules* to a complicated model to make it simpler and nevertheless keep some of the properties of the original system. A repeated application of reduction rules may in some cases even replace the reachability analysis as whole classes of Petri systems can be reduced to a sole transition.

The fundamental research of the reduction theory is Gérard Berthelot's doctoral dissertation [2], where he proves for several reduction rules that they preserve a set of properties of the original model. While Berthelot discussed the reduction theory of place/transition systems, Serge Haddad has presented three reduction rules for the regular systems [8], which are a special case of the coloured systems.

5.2 Reduction Rules

Reduction rules are here defined to be net transformation schemes which can be applied by inspecting solely the structure and the initial marking of the net. It is essential to rule out transformations which require a total reachability graph generation. It must immediately be stated that there does not and cannot exist any complete set of reduction rules that would make the reachability analysis unnecessary.

In the following, some of the most important reduction rules are presented for place/transition systems. It is assumed that $\langle P, T, W, M_0 \rangle$ is a P/T system as defined in Subsection 4.2.1. Excluding the removal of an irrelevant transition,

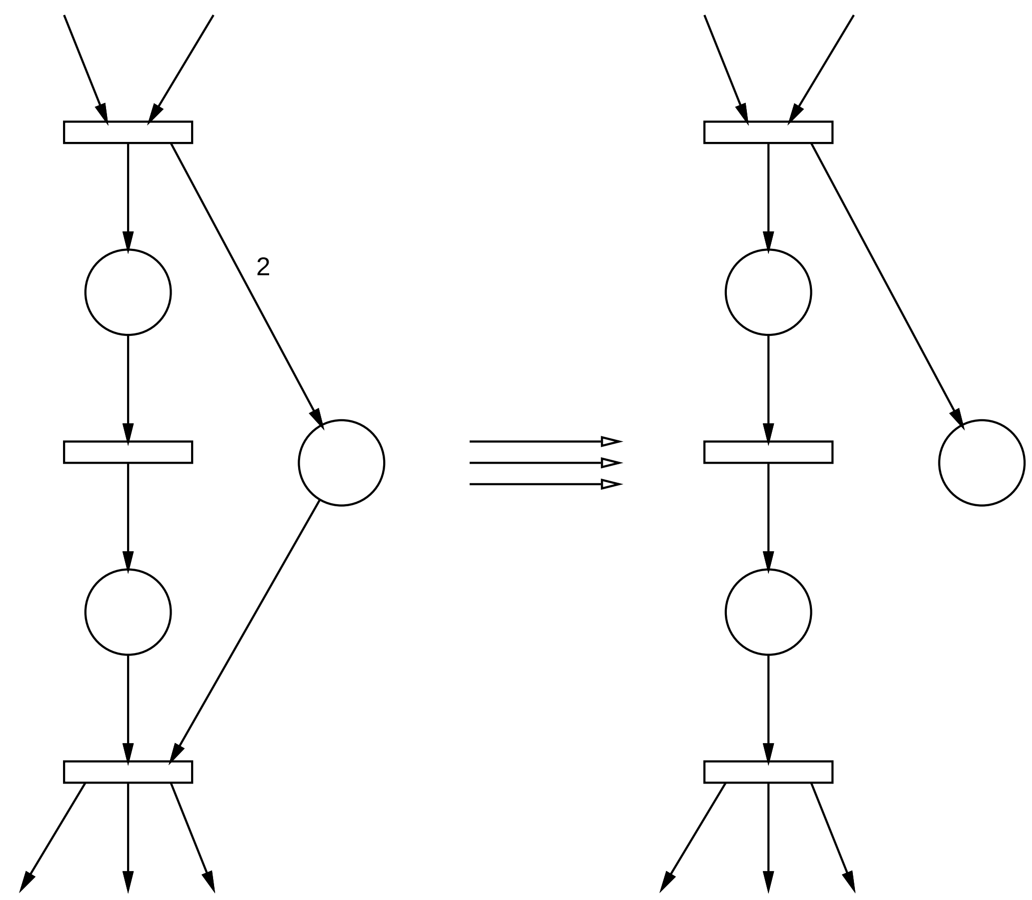


Figure 16: Simplification of an implicit place.

they all produce a system with equal liveness and boundedness properties. For a more perfect treatment, see [2].

Figure 16 represents an instance of the simplification rule of an *implicit place*. A place p is implicit if its marking is always at least a weighted sum of markings of a set Q of reference places and can never alone prevent a transition from firing:

$$\forall t \in T \; w(p)W(p,t) - \sum_{q \in Q} w(q)W(q,t) \leq w(p)M_0(p) - \sum_{q \in Q} w(q)W(q,t)$$

and

$$\forall t \in T \; w(p)\Delta(t,p) \geq \sum_{q \in Q} w(q)\Delta(t,q),$$

where $w : \{p\} \cup Q \rightarrow Z^+$ and $\Delta(t,p) = W(t,p) - W(p,t)$.

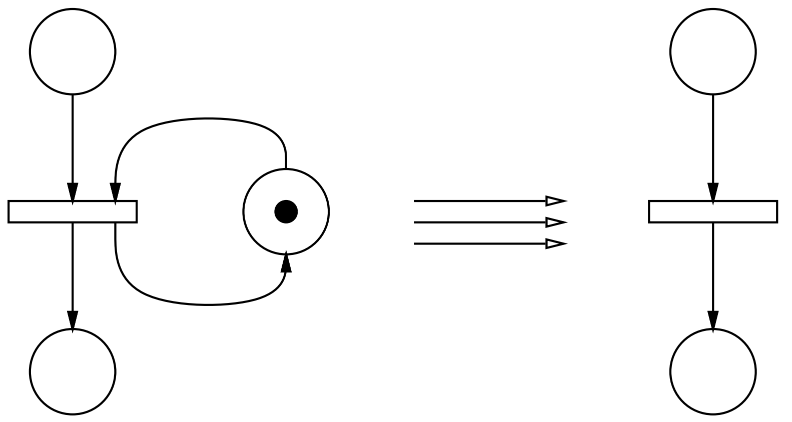


Figure 17: An implicit place with an empty reference set.

The arc weights $W(p,t)$ and $W(t,p)$ with which the implicit place p is associated can be changed so that $W(p,t) = 0$ and

$$W(t,p) = w(p)\Delta(t,p) - \sum_{q \in Q} w(q)\Delta(t,q).$$

Usually, $W(t,p)$ will become zero for all $t \in T$, in which case place p can be removed altogether.

The work of finding an implicit place in a net is combinatorially hard. Therefore it may be useful to limit oneself to some special cases like that of Figure 17 where the reference set Q is empty.

The simplification of removal of an implicit place has no effect on the dynamics of a system. Figure 18 represents a situation where of two transitions either both eventually fire or neither does. Therefore they could as well be represented by a single agglomerate transition.

The transitions around a place p are *postagglomerable* if

1. $M_0(p) = 0$,
2. $p\bullet \neq \emptyset$,
3. $\bullet p \cap p\bullet = \emptyset$,
4. $\forall t \in p\bullet \; \bullet t = \{p\}$ and
5. there exists a number $n \in N$ such that $\forall t' \in p\bullet \; W(p,t') = n$ and $\forall t \in \bullet p \; n | W(t,p)$,

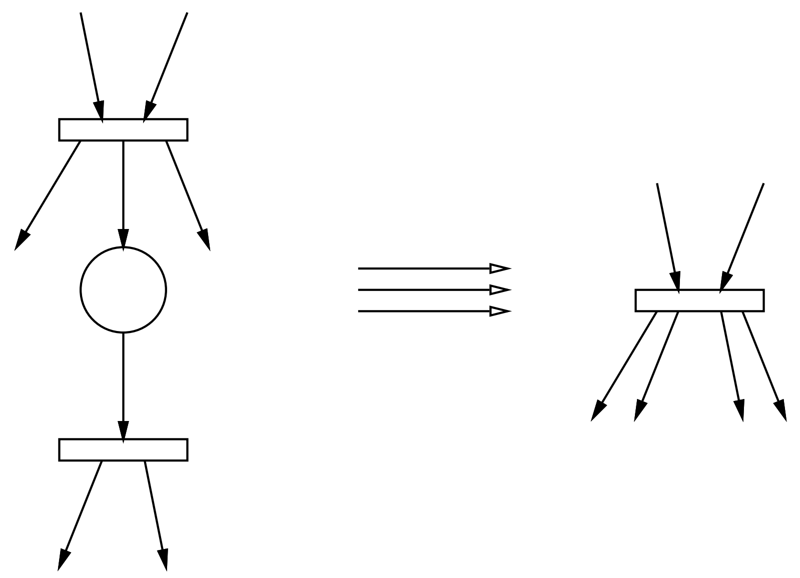


Figure 18: Postagglomeration of transitions.

where $m|n$ stands for "n is divisible by m". The postagglomeration of $\bullet p$ and $p\bullet$ then produces a net where p is removed and $\bullet p$ and $p\bullet$ are replaced by $\bullet p \times p\bullet$ and for all places q in $P \setminus \{p\}$,

$$\begin{aligned} W(q, \langle t, t' \rangle) &= W(q, t) \\ W(\langle t, t' \rangle, q) &= W(t, q) + \frac{W(t', q) W(t, p)}{W(p, t')}. \end{aligned}$$

The other agglomeration rule is the preagglomeration, which is seen in Figure 19. The transitions around a place p are *preagglomerable* if

1. $M_0(p) = 0$,
2. $\bullet p \cap p\bullet = \emptyset$,
3. $\forall t \in \bullet p \; t\bullet = \{p\}$,
4. $\forall t \in \bullet p \; \bullet t \neq \emptyset$,
5. $\forall t \in \bullet p \; \forall q \in \bullet t \; q\bullet = \{t\}$,
6. there exists a number $n \in N$ such that $\forall t \in \bullet p \; W(t, p) = n$ and $\forall t' \in p\bullet \; n | W(p, t')$.

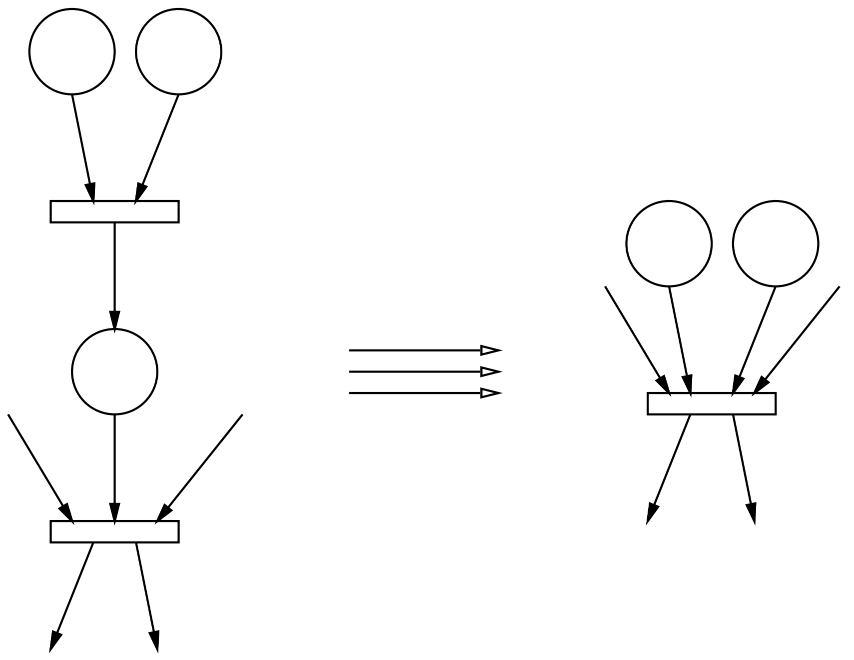


Figure 19: Preagglomeration of transitions.

The preagglomeration of $\bullet p$ and $p\bullet$ produces a net where p is removed and $\bullet p$ and $p\bullet$ are replaced by $\bullet p \times p\bullet$ and for all places q in $P \setminus \{p\}$,

$$\begin{aligned} W(q, \langle t, t' \rangle) &= \frac{W(q,t)W(p,t')}{W(t,p)} + W(q,t') \\ W(\langle t, t' \rangle, q) &= W(t', q). \end{aligned}$$

An *irrelevant transition* t has the property that

$$\forall p \in P \; \Delta(t, p) = 0,$$

so when it fires, the marking of the system will not change. An irrelevant transition may be removed, though the liveness of the system may not be the same, since it might be the only live transition of the system. Nevertheless, the boundedness of the system and the liveness of any single transition will not change.

Berthelot presented the rule of an irrelevant transition in [1] but omitted it in his thesis [2] probably because it may “kill” the system. However, the existence of any irrelevant transition somewhat slows up the generation of the reachability graph. And if the only liveness in a system is that of an irrelevant transition, it can even prevent the analyst from seeing the virtual deadlock.

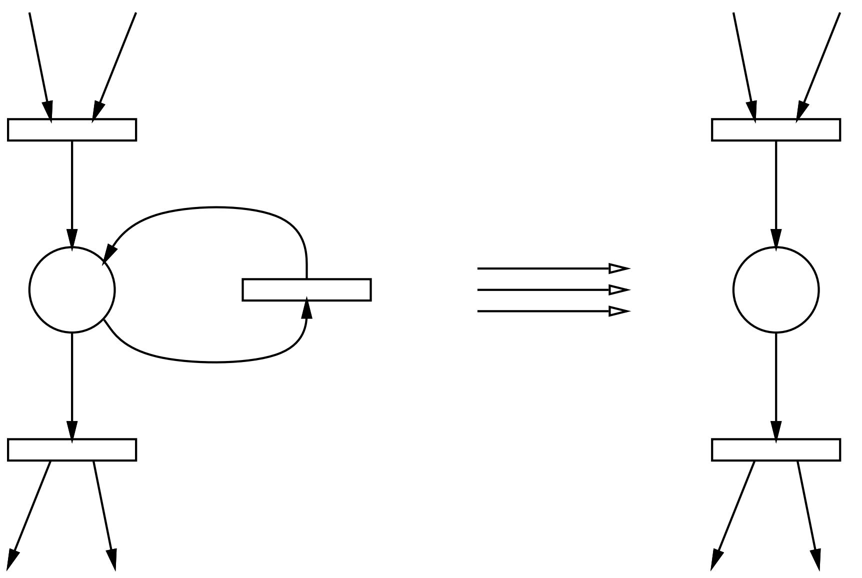


Figure 20: Removal of an irrelevant transition.

5.3 Example

As seen in Subsection 4.2.4, the dining philosopher model of Figure 21 has $3^n - 1$ states or markings. But according to the reduction theory, the emphasised transitions t_R and r_L can be (post- or pre-) agglomerated to one transition $\langle t_R, r_L \rangle$ (or rather, every instance of the pair $t_R[x]$, $r_L[x]$ is simultaneously agglomerable to $\langle t_R[x], r_L[x] \rangle$).

Agglomeration yields the system of Figure 22. Now $\langle t_R, r_L \rangle$ and r_R can be postagglomerated, which produces Figure 23.

In the model of Figure 23, the place P, which represents the hungry philosophers, can never prevent t_L from firing, so it is implicit. More formally, for each fixed x, if $Q = \{F[x]\}$ and $w(P[x]) = w(F[x]) = 1$, then for all transition instances t,

$$W(P[x], t) - W(F[x], t) \leq M_0(P[x]) - M_0(F[x])$$

or, since $M_0(P[x]) - M_0(F[x]) = 0$,

$$W(P[x], t) \leq W(F[x], t).$$

The low-level places $P[x]$ and $F[x]$ are namely handled by the transition instances $t_L[x]$, $\langle t_R, r_L, r_R \rangle[x]$ and $\langle t_R, r_L, r_R \rangle[(x-1) \bmod n]$. So

$$W(P[x], t_L[x])$$

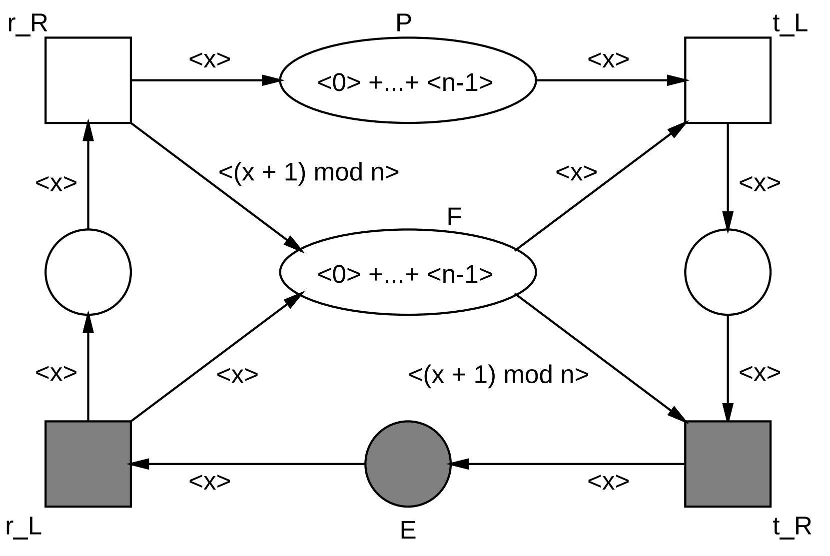


Figure 21: Redundant model of the dining philosophers.

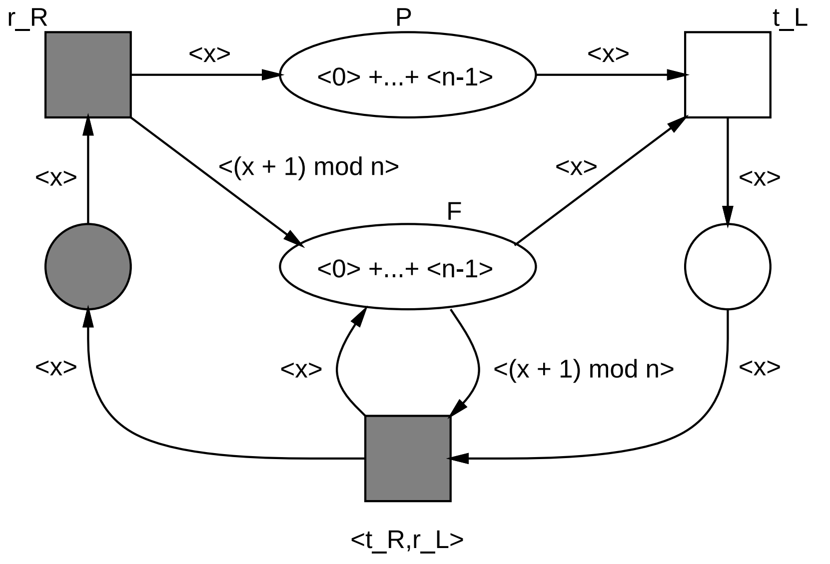


Figure 22: After (pre-) agglomeration.

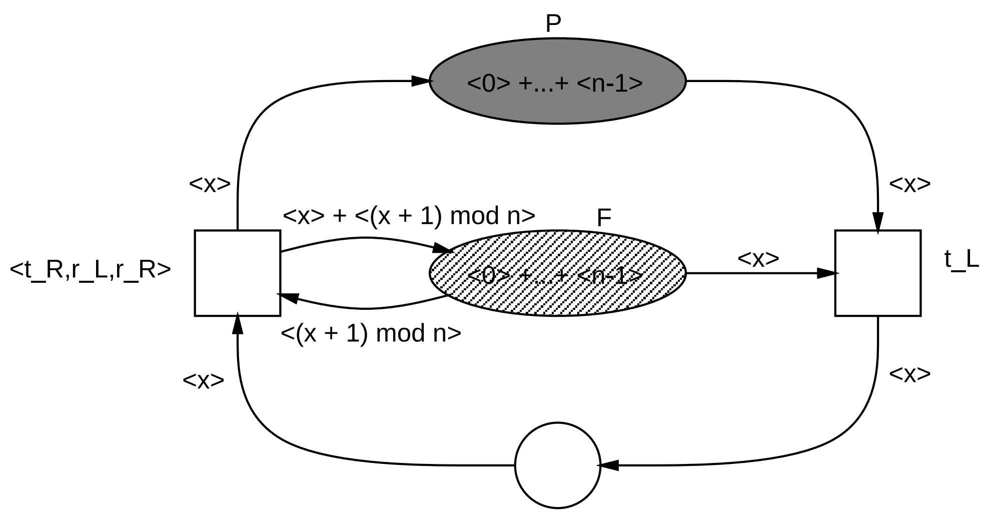


Figure 23: After postagglomeration.

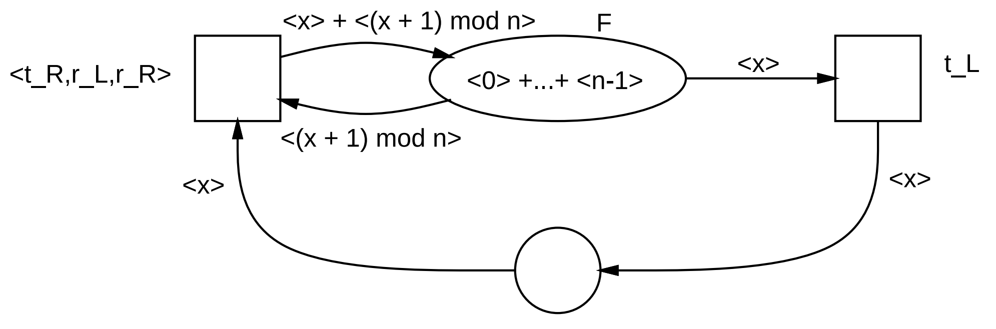


Figure 24: After removing the implicit place.

$$
\begin{aligned}
&= 1 \leq 1 = W(F[x], t_L[x]) \\
W&(P[x], \langle t_R, r_L, r_R\rangle[x]) \\
&= 0 \leq 0 = W(F[x], \langle t_R, r_L, r_R\rangle[x]) \\
W&(P[x], \langle t_R, r_L, r_R\rangle[(x-1) \bmod n]) \\
&= 0 \leq 1 = W(F[x], \langle t_R, r_L, r_R\rangle[(x-1) \bmod n]).
\end{aligned}
$$

Also,

$$\Delta(t, P[x]) \geq \Delta(t, F[x]),$$

as

$$
\begin{aligned}
\Delta&(t_L[x], P[x]) \\
&= 0 - 1 \geq 0 - 1 = \Delta(t_L[x], F[x]) \\
\Delta&(\langle t_R, r_L, r_R\rangle[x], P[x]) \\
&= 1 - 0 \geq 1 - 0 = \Delta(\langle t_R, r_L, r_R\rangle[x], F[x]) \\
\Delta&(\langle t_R, r_L, r_R\rangle[(x-1) \bmod n], P[x]) \\
&= 0 - 0 \geq 1 - 1 = \Delta(\langle t_R, r_L, r_R\rangle[(x-1) \bmod n], F[x]).
\end{aligned}
$$

The system of Figure 24 has obviously 2^n markings since each philosopher may be in either of the two places and all combinations are possible. So the state-space has been reduced to

$$1 : \frac{3^n - 1}{2^n} \approx 1 : 1.5^n$$

of the original. Thus the state-space is reduced exponentially with respect to n. On the other hand, the reduced system has exactly the same deadlock, in which all the philosophers take the left fork before anybody takes the right fork.

The stubborn set method produced even better results in Subsection 4.2.4. However, reduction is complementary to the stubborn set and symmetry method, too, so they should all be applied in order to obtain the best results.

5.4 Conclusion

In modelling and analysing, it is important to know the reduction theory. This is seen particularly well in the complex definitions of different reduction rules, especially the agglomerations. The analyst should be aware of the possible risks that lie in neglecting some odd-looking limitations. Their necessity is not very obvious but is revealed when the preservation of the different system properties is proved.

Reduction theory allows one to construct as detailled a model as the intuition requires. Then reachability analysis is made feasible by reducing the model with sound and possibly with non-sound transformations. Reduction rules are also simple to programme on a computer though most likely reduction is equally well performed by hand.

It is an interesting question whether a set of reduction rules has the Church-Rosser property, ie, whether different orders of applications always yield the same irreducible systems. It seems that the rules mentioned in Section 5.2 fulfil the property.

Reduction theory can be seen as a method of analysis competing with reachability analysis. It would, indeed, be a relief if reachability graph generation could be avoided totally. Berthelot has shown that the P/T systems that are bounded, live and persistent are fully reducible, ie, applying a set of reduction rules it is possible to transform such a system into one containing an isolated transition, which is trivially live. Such a set contains the simplification of implicit place, the agglomeration rules and some others that are not discussed above.

Unfortunately, the requirement of persistence is mostly not satisfied. (A system is persistent if an enabled transition can become disabled only by firing.) Therefore reduction, possibly together with other methods, usually forms a preliminary work for the reachability analysis.

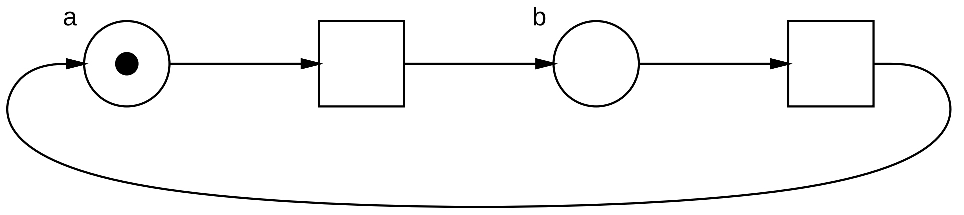


Figure 25: A simple elementary net system.

6 Other Approaches

6.1 Symbolic Model Checking

In most forms of reachability analysis, one, in a way or another, constructs the collection of reachable system states. The typical handling capability of a computer implementation is then from thousands to millions of states. But one can also view states and transitions as logical expressions and solving reachability and other problems as logical derivation. The approach is called *symbolic model checking*. In some cases, relatively short logical formulas may, owing to the regularities of the system, be able to capture immense state-spaces that would be intractable for the conventional method.

There are several, mainly modal, logics which are applied to symbolic model checking. Burch et al have presented a general framework under which the the others fall [6]. They also report of a case where they were able to generate a state-space of more than 10^{20} states!

The logic used by Burch et al is the μ-calculus, which can within the language express properties involving fixed points of logical formulas. Since the introduction of the μ-calculus would get a superfluously big proportion of this presentation and since the method of Burch et al at any rate transforms the formulas into Boolean expressions, the μ-calculus is here bypassed and the ideas are presented using Boolean or propositional logic by the aid of textual clarification.

Figure 25 represents a simple elementary net system. A case of the system is an element of $\{\perp, \top\} \times \{\perp, \top\}$. The set consisting of the initial case, can be given as the characteristic Boolean function

$$V_0(a, b) = a \wedge \neg b$$

and the events as the function

$$E(a, b, a', b') = a \oplus b \wedge a \oplus a' \wedge a' \oplus b'$$

(assuming that the exclusive or operation $\oplus$ has a higher precedence than $\wedge$). The reachability problem can be formulated in two ways. First, one can compute the function $Q(a,b)$ that represents the set of reachable markings. It must satisfy the following condition:

$$\begin{aligned} Q(a,b) &\equiv V_0(a,b) \vee \exists x,y(Q(x,y) \wedge E(x,y,a,b)) \\ &\equiv V_0(a,b) \vee (Q(\bot,\bot) \wedge E(\bot,\bot,a,b)) \vee (Q(\bot,\top) \wedge E(\bot,\top,a,b)) \\ &\quad \vee (Q(\top,\bot) \wedge E(\top,\bot,a,b)) \vee (Q(\top,\top) \wedge E(\top,\top,a,b)), \end{aligned}$$

ie, Q must be a fixed point of an expression. The interesting fixed point is the least one, or the most difficult to satisfy.

The least fixed point of Q is computed in a way which greatly resembles the ordinary reachability graph generation. One begins with $Q_0 = \bot$, or the empty set of cases. One then iteratively assigns

$$Q_{i+1}(a,b) = V_0(a,b) \vee \exists x,y(Q_i(x,y) \wedge E(x,y,a,b))$$

until $Q_{i+1} \equiv Q_i$. The computation will eventually halt since V, the set of possible cases, is finite and since the fixed point formula is *formally monotone* with respect to Q, ie, Q occurs only behind an even number of negations. Thus

$$\begin{aligned} Q_0 &= \bot \\ Q_1 &= (a \wedge \neg b) \vee \exists x,y(\bot \wedge x \oplus y \wedge x \oplus a \wedge a \oplus b) \\ &= a \wedge \neg b \\ Q_2 &= (a \wedge \neg b) \vee \exists x,y(x \wedge \neg y \wedge x \oplus y \wedge x \oplus a \wedge a \oplus b) \\ &= (a \wedge \neg b) \vee (\neg a \wedge b) \\ &= a \oplus b \\ Q_3 &= (a \wedge \neg b) \vee \exists x,y(x \oplus y \wedge x \oplus y \wedge x \oplus a \wedge a \oplus b) \\ &= (a \wedge \neg b) \vee (a \oplus b) \\ &= a \oplus b = Q_2. \end{aligned}$$

So $Q(a,b) = a \oplus b$.

The other approach is to compute the transitive closure Z of E. This is called *iterative squaring* by [6]. The fixed point equation is

$$\begin{aligned} Z(a,b,a',b') &\equiv E(a,b,a',b') \vee \exists x,y(Z(a,b,x,y) \wedge Z(x,y,a',b')) \\ &\equiv E(a,b,a',b') \\ &\quad \vee (Z(a,b,\bot,\bot) \wedge Z(\bot,\bot,a',b')) \\ &\quad \vee (Z(a,b,\bot,\top) \wedge Z(\bot,\top,a',b')) \\ &\quad \vee (Z(a,b,\top,\bot) \wedge Z(\top,\bot,a',b')) \\ &\quad \vee (Z(a,b,\top,\top) \wedge Z(\top,\top,a',b')) \end{aligned}$$

and the computation

$$
\begin{aligned}
Z_0 &= \bot \\
Z_1 &= (a \oplus b \wedge a \oplus a' \wedge a' \oplus b') \vee \exists x, y(\bot \wedge \bot) \\
&= a \oplus b \wedge a \oplus a' \wedge a' \oplus b' \\
Z_2 &= (a \oplus b \wedge a \oplus a' \wedge a' \oplus b') \\
&\quad \vee \exists x, y((a \oplus b \wedge a \oplus x \wedge x \oplus y) \wedge (x \oplus y \wedge x \oplus a' \wedge a' \oplus b')) \\
&= (a \oplus b \wedge a \oplus a' \wedge a' \oplus b') \vee (a \oplus b \wedge \neg(a \oplus a') \wedge a' \oplus b') \\
&= a \oplus b \wedge a' \oplus b' \\
Z_3 &= (a \oplus b \wedge a \oplus a' \wedge a' \oplus b') \vee \exists x, y((a \oplus b \wedge x \oplus y) \wedge (x \oplus y \wedge a' \oplus b')) \\
&= a \oplus b \wedge a' \oplus b' = Z_2.
\end{aligned}
$$

Now the set of cases reachable from the initial case is

$$
\begin{aligned}
Q(a, b) &= \exists x, y(V_0(x, y) \wedge Z(x, y, a, b)) \\
&= \exists x, y(x \wedge \neg y \wedge x \oplus y \wedge a \oplus b) \\
&= a \oplus b.
\end{aligned}
$$

Iterative squaring has the advantage that it does not refer to the initial case, so the system can easily be parameterised with it. The method assumes that the path lengths of events are about the same throughout the state-space.

Since the composition of events simultaneously takes place all over the net, iterative squaring needs only a logarithmic number of iterations with respect to the "normal" computation of Q. This does not, however, necessarily mean a smaller computation time and the needed computation space is probably bigger.

The symbolic state-space generation seems rather complicated even in the extremely simple example. For instance, the evaluation of the existential quantifier $\exists$ requires the traversal of all the different combinations. This problem, however, is not so bad. A tree algorithm similar to some algorithms of the earlier chapters can be applied to avoid going through every combination. The order of quantified variables then plays a crucial role.

How are then the formulas reduced and how are they compared in order to halt the iteration? The normal form used by Burch et al for the Boolean expressions is the so-called *binary decision diagram* (see Figure 26), a labelled directed acyclic graph having a unique minimal form under a given order of variables. The computation of the minimal form is inexpensive and the minimal forms make the task of comparison easy. The BDD's for Q_i and Z_i are presented in Figure 27.

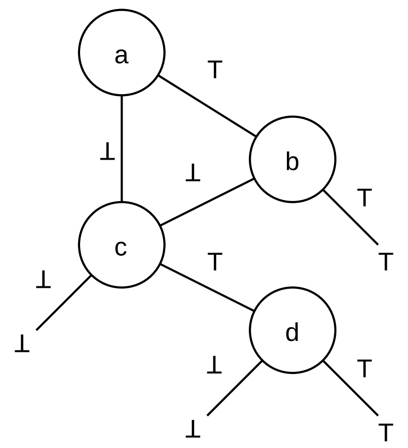


Figure 26: The binary decision diagram of the propositional formula $(a \wedge b) \vee (c \wedge d)$.

It is very difficult to estimate when symbolic model checking is profitable. Burch et al briefly present a case of a simple synchronous three-address ALU pipeline with exclusive or and addition operations. One configuration produces 41 000 nodes into the BDD in 22 minutes on a Sun 3 workstation. The system has about $5 \cdot 10^{20}$ states, which is far beyond the reach of the conventional reachability analysis. On the other hand, an ALU with multiplication is known not to have a satisfactory BDD.

If a system has regularities, the languages of logic can surely express them best. However, their disadvantage with respect to ad hoc methods is the difficulty of dealing with symbols (the NP-completeness). It is stated in [6] that "the BDD method for representing state sets [eg] in the CTL model checking problem is only of heuristic value and does not improve the asymptotic complexity of model checking". One may suspect that one seldom falls on a situation where symbolic model checking beats the conventional reachability analysis, especially considering those cases where Petri systems are appropriate, ie, where the events do not have such convenient properties as addition and exclusive or do.

6.2 Incomplete Reachability Analysis

The conventional state-space generation constructs a collection of states and mostly also the state transitions (the reachability graph). Computer implementations gather the states into some data structure where the generated states can be searched as quickly as possible, since for every new state the programme must check whether the state has already been generated. This

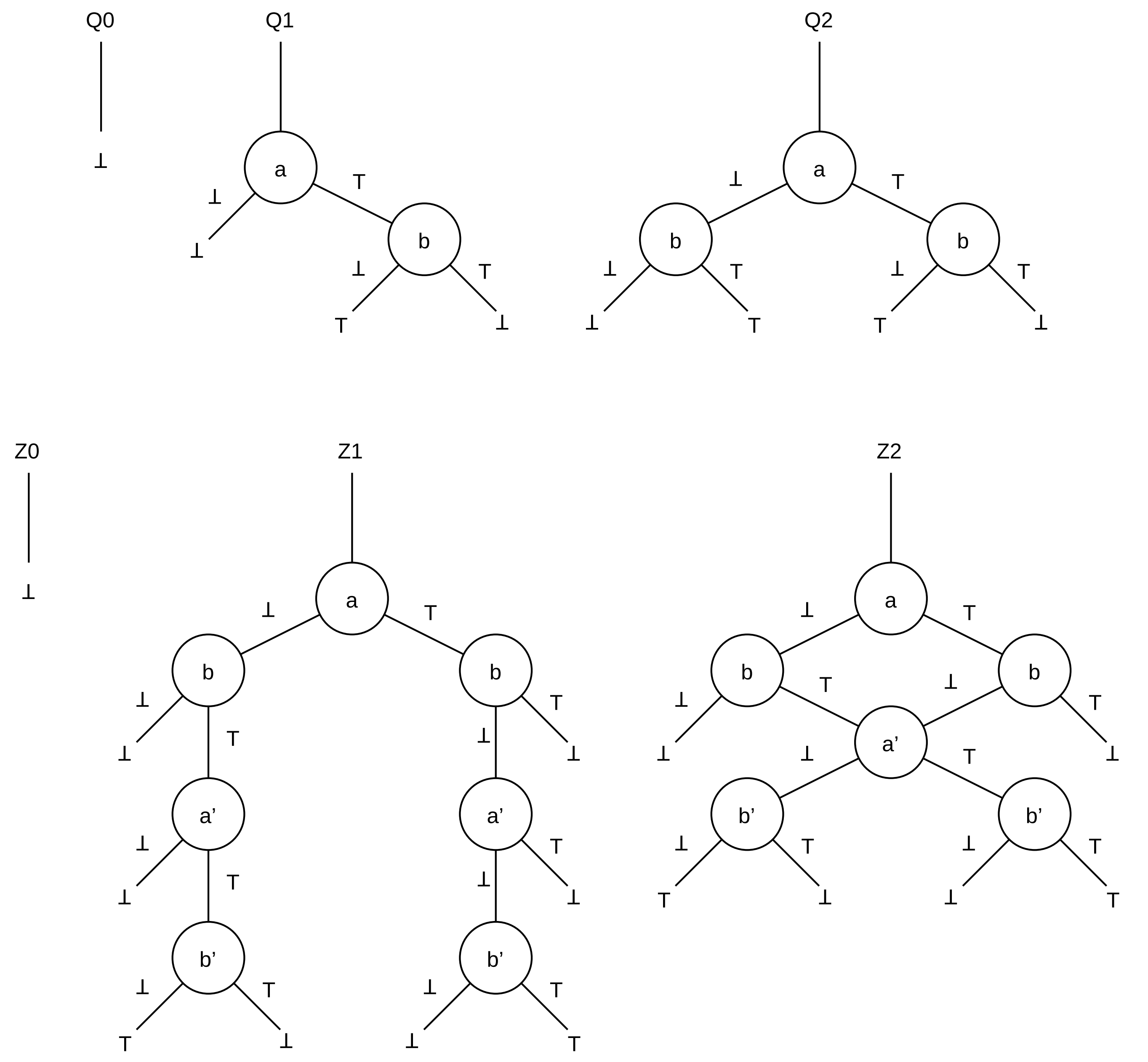


Figure 27: The BDD's for Q_i and Z_i with $a \prec b \prec a' \prec b'$.

is accomplished by means of a hash function which maps the states very uniformly to a range of cells. Every cell is a dynamic data structure (list or tree) containing all the states that are assigned to the cell. The least premise is to keep all the states in the memory. A typical state will take some hundreds or thousands of bits of memory.

Gerard Holzmann has proposed that the completeness of the state-space generation could be sacrificed in return for the increase of the number of states and speed of generation by orders of magnitude [9, 10]. In the method, each hash table cell is a bit originally containing zero. States are not recorded (except the front of the generation). Each generated state is hashed and the corresponding hash table cell is checked. If it is zero, the state is a new one and the cell bit is raised to one and the state is added to those whose successor states are to be generated. If the cell is one, the state *may have been* generated previously, so the successors of the state are not generated.

If the size of the front states is relatively small compared with the number of all states, Holzmann's method can handle several hundred times more states. The maximum number of states is even known beforehand (the size of the hash table in bits) and the programme will never run out of memory. Also, determining whether a state is new (checking a bit) is much faster than comparing states (though not necessarily so much as Holzmann suspects since states need not be gathered in lists but rather in trees, and the first bit or bits of two states usually resolves the equality).

The incompleteness of the method may seem suspicious but if a good hash function is chosen and size of the complete state-space is at most approximately the same as the number of hash table cells, only a few percent of the states are lost. Then it is highly probable that any errors are detected all the more as most errors repeat themselves all over the state-space. The result can be improved by rerunning with a new hash function. In any case, the results are more complete than the exhaustive search methods which are able to handle only a small portion of the whole state-space. Many conventional implementations simply crash without giving any results.

Holzmann reports to have applied the method to verifying the specification of a real switch. The computer generated 250 million states in an hour of CPU time. Still he could only analyse parts of the specification: "practical" systems are usually designed to contain an infinity of states.

There are problems with the method, though. First, it does not store any of the state transitions, which severely limits the properties that can be examined. On the other hand, the existence of deadlocks seems to be the most important question that can be decided fully automatically. Holzmann does not report, either, how much memory space should be estimated for the front states. If half of the memory were given to the front, it would correspond to an estimation of approximately 0.1% for the front/total ratio.

Holzmann's method is a simple but brilliant idea for managing huge state-spaces. It is suitable as well for Petri systems as for any other finite state machine formalism. Despite its drawbacks, the method may often be the only possible way to verify large systems. Since it is also easy to implement, it should be an option in any automatic reachability analyser.

7 Conclusion

7.1 Evaluation of the Methods

Here is a brief summary of the reviewed methods.

The symmetry method has both a clear intuition and strong mathematical basis, so it looks like a good choice for reducing reachability graph generation. It produces a compact representation of the ordinary reachability graph. It needs the help of the analyst in detecting the symmetries but they are often obvious and their soundness can be verified automatically. However, the time saved by the method is of the same magnitude as the time spent in computing the symmetries, so the possible profits of the method are in space savings, which may be considerable, and also in computing the symmetries if it is nevertheless faster by, say, some constant factor. One can still doubt whether such complete symmetries necessary for the method occur in real systems.

The worst problem of the parameterisation method is that it has not been defined what the exact relationship between the ordinary and parameterised reachability graph is. In addition, the extremely complex computation probably makes the method slower than the ordinary state-space generation. Therefore, at least at this stage of the theory, parameterised reachability graph generation is not of much value.

The stubborn set method is based on an interesting idea, which covers a good part of the whole theory of priorities. Excellent algorithms have also been presented. If the practical experiments confirm the expectations, the method ought to become one of the basic tools of most reachability analysis systems. The original form of stubbornness aims at finding all deadlock states. New papers have extended the application area a little but the more information of the ordinary reachability graph is wanted to be preserved, the less the stubborn set method is able to reduce.

The reduction theory is a miscellaneous collection of model transformation rules. Each rule has been proved to preserve some properties of the original model. Application of reduction rules is typically inexpensive since they are usually defined in terms of the structure of the model and applied before the reachability graph generation. It seems that practical models tend to have redundancy that can be got rid of with the existing reduction rules. However, the reduction theory cannot usually remove the ordinary reachability problem, so other methods are needed.

Symbolic model checking can be seen as a kind of generalisation of methods like symmetry and parameterisation method. Formal logic is capable of expressing and managing a vast variety of regularities. Therefore, despite it cannot be combined well with other methods, it in a way contains them all. State-spaces of astronomical sizes that have been analysed successfully by symbolic model checking have been reported. However, it may be conjectured that most distributed systems are likely to be more difficult to manage and in most practical cases symbolic model checking will be computationally unfeasible.

Incomplete reachability analysis is a simple but valuable idea which is at its best when the ordinary reachability graph generation is not feasible, and this may be the usual case. It has the same problem as the stubborn set method: it is only good at finding deadlocks. But on the other hand, existence of deadlocks is an important property to examine.

7.2 Combining Methods

It turns out that most of the presented methods are complementary. Carefully used, they can yield better results when used together than alone.

The combination of symmetry method and stubborn set method has been tried out before [18, 22]. They can both drop the exponential state-space of the philosopher example to quadratic but to linear when used together as they deal with different aspects of a reachability graph.

Reduction theory, especially the agglomeration rules, overlaps the area of stubborn set method. However, it can well be added to the symmetry-stubborn-set combination, since reduction rules are applied before reachability graph generation and the stubborn set method and the methods do have fully independent parts.

Incomplete reachability analysis can also be used to augment the set of methods. Then the other methods may be seen as auxiliary methods for the incomplete reachability analysis. First, reduction theory is applied to minimise the potential state-space. Then incomplete reachability analysis is carried out by ruling out all symmetrical and non-stubborn transitions. This procedure leaves more holes to the hash table and the limited memory space can be used more efficiently for irredundant states. A practical reachability analysis system would thus contain different methods which can be included in or excluded out of the analysis as seen appropriate.

References

[1] Berthelot, G. and Roucairol, G.: *Reduction of Petri-Nets.* Mathematical Foundations of Computer Science 1976, Lecture Notes in Computer Science 45, Springer-Verlag, Berlin 1976, pp 202–209.

[2] Berthelot, G.: *Transformations et analyse de réseaux de Petri, application aux protocoles.* Thèse d'état, Université P. et M. Curie, Paris 1983, 417 p.

[3] Berthelot, G.: *Transformations and Decompositions of Nets.* Petri Nets: Central Models and Their Properties, Lecture Notes in Computer Science 254, Springer-Verlag, Berlin 1987, pp 359–376.

[4] Brams, G.W.: *Réseaux de Petri: théorie et pratique*, Tome 1, Théorie et analyse, Masson Paris 1983, 184 p.

[5] Brams, G.W.: *Réseaux de Petri: théorie et pratique*, Tome 2, Modélisation et applications, Masson Paris 1983, 160 p.

[6] Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L. and Hwang, L.J.: *Symbolic Model Checking:* 10^{20} *States and Beyond.* Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, Los Alamitos CA 1990, pp 428–439.

[7] Colom, J., Martinez, J. and Silva, M.: *Packages for Validating Discrete Production Systems Modeled with Petri Nets.* Proceedings of the IMACS-IFAC Symposium, Villeneuve d'Ascq 1986, pp 457–462.

[8] Haddad, S.: *Generalization of Reduction Theory to Coloured Nets.* Proceedings of the Ninth European Workshop on Application and theory of Petri Nets, Vol II, Venice 1988, 18 p.

[9] Holzmann, G.J.: *On Limits and Possibilities of Automated Protocol Analysis.* Proceedings of Protocol Specification, Testing and Verification: VII, Zürich 1987, 7 p.

[10] Holzmann, G.J.: *Algorithms for Automated Protocol Verification.* AT&T Technical Journal, Jan/Feb 1990, pp 32–44.

[11] Huber, P., Jensen, A.M., Jepsen, L.O. and Jensen, K.: *Towards Reachability Trees for High-level Petri Nets.* Technical report DAIMI PB-174, Department of Computer Science, Aarhus University, Aarhus 1985, 19+42 p.

[12] Lindqvist, M.: *Parameterized Reachability Trees for Predicate/Transition Nets.* Doctoral thesis, Acta Polytechnica Scandinavica, Mathematics and Computer Science Series No 54, Helsinki 1989, 120 p.

[13] Lindqvist, M.: *Parameterized Reachability Trees for Predicate/Transition Nets.* Proceedings of the Eleventh International Conference on Application and Theory of Petri Nets, Paris 1990, pp 22–42.

[14] Overman, W.T.: *Verification of Concurrent Systems: Function and Timing.* PhD thesis, University of California, Los Angeles, Los Angeles CA 1981, 174 p.

[15] Pnueli, A.: *Applications of Temporal Logic to the Specification and Verification of Reactive Systems: Survey of Current Trends.* Current Trends in Concurrency, Overviews and Tutorials, Lecture Notes in Computer Science 224, Springer-Verlag, Berlin 1986, pp 510–584.

[16] Reisig, W.: *Petri Nets. An Introduction.* Springer-Verlag, Berlin 1982, 161 p.

[17] Sedgewick, R.: *Algorithms.* Addison-Wesley, Reading MA 1983, 551 p.

[18] Valmari, A.: *Error Detection by Reduced Reachability Graph Generation.* Proceedings of the Ninth European Workshop on Application and Theory of Petri Nets, Venice 1988, pp 95–112.

[19] Valmari, A.: *Heuristics for Lazy State Generation Speeds up Analysis of Concurrent Systems.* Mäkelä, M., Linnainmaa, S. and Ukkonen, E. (ed.), Proceedings of the Finnish Artificial Intelligence Symposium (Suomen tekoälytutkimuksen päivat), Vol 2, Helsinki 1988, pp 640–650.

[20] Valmari, A.: *State Space Generation: Efficiency and Practicality.* Doctoral thesis, Tampere University of Technology Publications 55, Tampere 1988, 170 p.

[21] Valmari, A.: *Eliminating Redundant Interleavings during Concurrent Program Verification.* Proceedings of PARLE 89, Volume II, Lecture Notes in Computer Science 366, Springer-Verlag, Berlin 1989, pp 89–103.

[22] Valmari, A.: *Stubborn Sets for Reduced State Space Generation.* Supplement to the Proceedings of the Tenth International Conference on Application and Theory of Petri Nets, Bonn 1989, pp 1–22.

[23] Valmari, A.: *A Stubborn Attack on State Space Explosion.* Proceedings of Workshop on Computer-Aided Verification, Vol I, DIMACS Technical Report 90-31, New Brunswick NJ 1990, 15 p.