EFFICIENT DETECTION OF DEADLOCKS IN PETRI NETS

Licentiate's thesis
Kimmo Varpaaniemi

September 15, 1993

Helsinki University of Technology
Faculty of Information Technology
Department of Computer Science
Digital Systems Laboratory

Supervisor: Professor Leo Ojala

Instructor: Assistant Professor Mikko Tiusanen

HELSINKI UNIVERSITY OF TECHNOLOGY **ABSTRACT OF THE LICENTIATE'S THESIS**

Author: Kimmo Varpaaniemi

Name of the Thesis:
Efficient Detection of Deadlocks in Petri Nets

Name in Finnish:
Petri-verkkojen lukkiutumien tehokas etsintä

Date: September 15, 1993	**Pages:** vii+63
Faculty: Information Technology	**Professorship:** Tik-79 Digital Systems
Supervisor: Prof. Leo Ojala	**Instructor:** Ass. Prof. Mikko Tiusanen

Abstract: *Reachability analysis* (exhaustive simulation, state space generation) is a powerful formal method for analysis of concurrent and distributed finite state systems. It suffers from the *state space explosion problem*, however: the state space of a system can be far too large to be completely generated even if finite. This thesis considers two promising methods, Valmari's *stubborn set method* and Godefroid's *sleep set method*, to avoid generating all of the state space when searching for *undesirable reachable terminal states*, also called *deadlocks*. What makes deadlocks especially interesting is the fact that the verification of a *safety property* can often be reduced to deadlock detection. The considered methods utilize the *independence of transitions* to cut down on the number of states inspected during the search. These methods have been *combined* by Godefroid, Pirottin, and Wolper to further reduce the number of inspected states. All of these methods guarantee that all reachable terminal states are found if the complete state space is finite.

Petri nets are a widely used model for concurrent and distributed systems. This thesis shows that the stubborn set method and the sleep set method can be combined without any of the assumptions previously placed on the stubborn sets as far as the detection of reachable terminal states in *place/transition nets*, a class of Petri nets, is concerned. The obtained result is actually more general and gives a sufficient condition for a method to be compatible with the sleep set method in the detection of reachable terminal states in place/transition nets.

The number of enabled transitions in a stubborn set can drastically affect the number of states inspected by the stubborn set method during the search for reachable terminal states. This work presents some heuristics for relieving the problem.

This thesis emphasizes the value of *dynamically stubborn sets* as a useful generalization of stubborn sets and shows some results that improve the understanding of the stubborn set method.

Keywords: reachability analysis, Petri nets, deadlocks, stubborn set method, sleep set method

TEKNILLINEN KORKEAKOULU **LISENSIAATINTYÖN TIIVISTELMÄ**

Tekijä: Kimmo Varpaaniemi

Työn nimi: Efficient Detection of Deadlocks in Petri Nets

Suomenkielinen nimi: Petri-verkkojen lukkiutumien tehokas etsintä

Päivämäärä: 15. syyskuuta 1993	**Sivumäärä:** vii+63
Osasto: Tietotekniikka	**Professuuri:** Tik-79 Digitaalitekniikka
Valvoja: prof. Leo Ojala	**Ohjaaja:** vt.apul.prof. Mikko Tiusanen

Tiivistelmä: *Saavutettavuusanalyysi* (täydellinen simulointi, tila-avaruuden generointi) on voimakas formaali menetelmä äärellistilaisten rinnakkaisten ja hajautettujen järjestelmien analysointiin. Saavutettavuusanalyysin käytettävyyttä rajoittaa kuitenkin ns. *tila-avaruuden räjähdysongelma*: järjestelmän tila-avaruus voi olla niin suuri, että käytettävissä olevat resurssit eivät riitä koko tila-avaruuden generointiin.
Tässä työssä tarkastellaan kahta lupaavaa menetelmää järjestelmän mahdollisten *lukkiutumien* eli alkutilasta saavutettavissa olevien epäsuotuisien seuraajattomien tilojen löytämiseksi ilman koko tila-avaruuden läpikäyntiä: Valmarin *itsepäisten joukkojen menetelmä* ja Godefroid'n *unijoukkojen menetelmä* (engl. *sleep set method*). Lisäksi tarkastellaan näiden kahden menetelmän *yhdistämistä*, lähtökohtana Godefroid'n, Pirottinin ja Wolperin esittämät tulokset. Lukkiutumien löytämisen ongelma on keskeinen ongelma mm. siksi, että monen ns. *turvallisuusominaisuuden* voimassaolon selvittämisen ongelma on muunnettavissa lukkiutumien löytämisen ongelmaksi. Itsepäisten joukkojen menetelmä ja unijoukkojen menetelmä käyttävät kumpikin hyväksi sitä, että tapahtumien järjestyksellä ei ole läheskään aina merkitystä lukkiutumien löytämisen kannalta. Kumpikin menetelmä samoin kuin niiden yhdistelmä löytää kaikki alkutilasta saavutettavissa olevat seuraajattomat tilat.
Petri-verkkoja käytetään usein mallittamaan rinnakkaisia ja hajautettuja järjestelmiä. *Paikka-siirtymäverkot* ovat yksi Petri-verkkojen luokka. Tässä työssä osoitetaan, että itsepäisten joukkojen menetelmä ja unijoukkojen menetelmä voidaan, sikäli kuin kyseessä on alkutilasta saavutettavissa olevien seuraajattomien tilojen etsintä paikka-siirtymäverkossa, yhdistää ilman, että käytettävien itsepäisten joukkojen tarvitsisi olla aiemmissa yhdistelmämenetelmän versioissa esitettyjen oletusten mukaisia. Varsinainen osoitettu tulos on vieläpä jonkin verran yleisempi ja määrittelee ehdon, jonka toteuttava menetelmä sopii yhteen unijoukkojen menetelmän kanssa etsittäessä alkutilasta saavutettavissa olevia seuraajattomia tiloja paikka-siirtymäverkossa.
Se, kuinka monta vireessä olevaa transitiota käytettävässä itsepäisessä joukossa on, voi vaikuttaa varsin ratkaisevalla tavalla siihen, kuinka monta tilaa joudutaan tutkimaan koko haun aikana. Tässä työssä esitetään heuristiikkoja tämän ongelman lieventämiseksi.
Tässä työssä myös korostetaan ns. *dynaamisesti itsepäisten joukkojen* arvoa itsepäisten joukkojen yleistyksenä sekä osoitetaan joukko tuloksia, jotka edesauttavat itsepäisten joukkojen menetelmän ymmärtämistä.

Avainsanat: saavutettavuusanalyysi, Petri-verkot, lukkiutumat, itsepäisten joukkojen menetelmä, unijoukkojen menetelmä

Preface

This licentiate's thesis has been carried out in the Digital Systems Laboratory of Helsinki University of Technology. I am grateful to the supervisor of this thesis, Professor Leo Ojala, and to the instructor of this thesis, Assistant Professor Mikko Tiusanen, for their continuous support. In addition, I would like to thank Assistant Professor Antti Valmari and Lic.Tech. Marko Rauhamaa for fruitful discussions on the subject of this thesis.

The financial support received from the Emil Aaltonen Foundation is also gratefully acknowledged.

Espoo, September 15, 1993

Kimmo Varpaaniemi

Contents

List of Symbols

Abbreviations

AV	an abbreviation of the name Antti Valmari, used as a prefix
AVSD	the principle of AV-strong dynamic stubbornness
BDD	a binary decision diagram
CD	the principle of conventional dynamic stubbornness
D1	the first principle of dynamic stubbornness
D2	the second principle of dynamic stubbornness
iff	if and only if
PE	the principle of persistence and conditional stubbornness
SD1	the first principle of strong dynamic stubbornness
SD2	the second principle of strong dynamic stubbornness

Logic

$\neg$	negation
$\wedge$	conjunction
$\vee$	disjunction
$\Rightarrow$	implication
$\Leftrightarrow$	equivalence
$\forall$	the universal quantifier
$\exists$	the existential quantifier

Set Theory

$\emptyset$	the empty set
$a \in A$	a is a member of the set A
$a \notin A$	a is not a member of the set A
$A \subseteq B$	the set A is a subset of the set B
$A \cap B$	the intersection of the sets A and B
$A \cup B$	the union of the sets A and B
$A \setminus B$	the complement of the set B with respect to the set A
2^A	the power set (the set of subsets) of the set A
$A \times B$	the Cartesian product of the sets A and B
$(A \to B)$	the set of functions from the set A to the set B
$\bigcup_{a \in A} B(a)$	$\{b \mid \exists a \in A\ b \in B(a)\}$, where B is a function from the set A to some class of sets
$\|A\|$	the size of the finite set A

Numbers

$\sum_{i=1}^{k} n_i$	the sum $n_1 + n_2 + \ldots + n_k$
ω	a formal infinite number, $\omega \notin N$
N	the set of natural numbers, including 0
N_ω	$N \cup \{\omega\}$

Place/Transition Nets: Structure

${}^\bullet x$	the set of input elements of x, where x is a place or a transition
$x^\bullet$	the set of output elements of x, where x is a place or a transition
ε	the empty transition sequence
σ	a transition sequence, finite or infinite
$\varsigma(\sigma, n)$	the prefix of length n of an infinite transition sequence σ
τ	a transition
F	the set of arcs of a net
K	the capacity function of a net
$\mathcal{M}$	the set of markings of a net
M	a marking
$M(s)$	the number of tokens in s at M
M_0	the initial marking of a net
S	the set of places of a net
s	a place
T	the set of transitions of a net
t	a transition
T^*	the set of finite transition sequences of a net
T_s	a subset of T
T_s^*	the set of finite sequences of transitions in T_s
W	the arc weight function of a net

Place/Transition Nets: Behaviour

$M[\sigma\rangle$	σ is enabled at M
$M[\sigma\rangle_f$	σ is f-enabled at M
$M[\sigma\rangle M'$	a finite transition sequence σ leads (can be fired) from M to M'
$M[\sigma\rangle_f M'$	a finite transition sequence σ f-leads (can be f-fired) from M to M'
$\eta(\sigma, M)$	the set of alternative sequences of a finite transition sequence σ at M
$\vartheta(\sigma, M)$	the set of length-secure alternative sequences of a finite transition sequence σ at M
$\iota(\sigma, M)$	the conditional trace of a finite transition sequence σ at M

$\pi(\sigma, M)$	the set of enabled permutations of a finite transition sequence σ at M
G	the set of scapegoat generators of a net

Stubborn Set and Sleep Set Algorithms

φ	a truth-valued function on $\mathcal{M} \times T \times T \times 2^T$
$E_1(M, s)$	the set of transitions that could increase the number of tokens in s and are not disabled by s at M
$E_2(M, t, s)$	the set of transitions that could decrease the number of tokens in s or get disabled because of the firing of t at M, $E_4(s) \subseteq E_2(M, t, s)$
$E_3(M, t, s)$	the set of transitions that are not disabled by s at M and could increase the number of tokens in s or deposit more tokens to s than t, $E_1(M, s) \subseteq E_3(M, t, s)$
$E_4(s)$	the set of transitions that could decrease the number of tokens in s
$E_{14}(f, b, M, t)$	the set of transitions immediately succeeding t in the $\langle f, b\rangle$-dependency graph at M
$E^*_{14}(f, b, M, t)$	the set of transitions accessible from t in the $\langle f, b\rangle$-dependency graph at M
$R_{14}(f, b, M)$	the set of edges of the $\langle f, b\rangle$-dependency graph at M
T_0	a subset of T
V_{AND}	the set of and-vertices of an and/or-graph
V_{OR}	the set of or-vertices of an and/or-graph

Others

□	the end delimiter of the proof of a lemma or a theorem

List of Figures

1 Introduction

Concurrent and distributed systems such as telecommunication protocols and process control systems influence and affect the lives of millions of people daily all over the world today. The design of these systems often involve so difficult problems related to timing that the traditional testing and analysis methods are not adequate. One possible solution to this problem is the discerning use of appropriate *formal methods*.

Reachability analysis, also known as *exhaustive simulation* or *state space generation*, is a powerful formal method for detecting errors in such concurrent and distributed systems that have a finite state space. It suffers from the so called *state space explosion problem*, however: the state space of the system can be far too large with respect to the time and other resources needed to inspect all states in the space. Fortunately, errors such as *undesirable reachable terminal states*, also called *deadlocks*, can be detected in a variety of cases without inspecting all reachable states of the system. What makes deadlocks especially interesting is the fact that the verification of a *safety property* can often be reduced to the detection of deadlocks, as shown by Godefroid and Wolper [30] among others.

Petri nets [8, 43, 64] are a widely used model for concurrent and distributed systems. This thesis concentrates on the problem of detecting *reachable terminal states* in *place/transition nets*, a class of Petri nets. Two promising methods are studied: Valmari's *stubborn set method* [70, 71, 73, 74, 75, 76, 78, 79, 80] and Godefroid's *sleep set method* [25, 27, 28, 30, 26, 36, 86, 87]. Both methods utilize the *independence of transitions* to cut down on the number of states inspected during the search. These methods have also been *combined* by Wolper and Godefroid [86], Godefroid and Pirottin [28], and Wolper, Godefroid, and Pirottin [87] to further reduce the number of inspected states. All of these methods guarantee that all reachable terminal states are found if the complete state space is finite. Though we shall concentrate on place/transition nets, we shall also make some remarks concerning *high-level nets* [8, 43]. Place/transition nets of actual systems tend to be very large. On the other hand, using high-level nets one can make compact models in a natural way. Fortunately, a high-level net can often be *unfolded* into a behaviourally equivalent finite place/transition net, and, using the inverse mapping of the unfolding mapping, the place/transition net can be folded back into the high-level net [24, 40]. This provides a path of extending results on place/transition nets to high-level nets.

The application of the stubborn set method and the sleep set method is not limited to Petri nets. The methods have been applied to several models of concurrency by Valmari [75, 79], Godefroid and Pirottin [28], Wolper and Godefroid [86], and Peled [59] among others. Valmari [71], and Godefroid and Kabanza [27] have presented how the stubborn set method and the sleep set method can be used to improve the *graph search methods* used for artificial intelligence. The independence of rules in production systems of artificial intelligence [56] resembles the independence of actions in concurrent and distributed systems in many senses. The essential point in the stubborn set method and the sleep set method is that they utilize the independence of actions or rules. *Refined independence relations* are important since the more refined is the independence relation the less states usually have to be inspected. Best and Lengauer [5], and Katz and Peled [46] among others have studied refined independence relations and developed general concepts of independence.

The application of the stubborn set method and the sleep set method is not limited to the detection of reachable terminal states either. Both methods can be extended to verify properties expressed as linear temporal logic formulae without a next state -operator, as shown by Valmari [76, 79], Godefroid and Wolper [29], and Peled [59].

The rest of this chapter has been organized as follows: reachability analysis is presented in Section 1.1. Section 1.2 presents the stubborn set method and the sleep set method. In Section 1.3 we give a brief description of the contributions of this thesis. Related work is considered in Section 1.4. Section 1.5 shows how the rest of this thesis has been organized.

1.1 Reachability Analysis

A classic way to detect errors in a system is *testing.* However, it is often difficult to test the system sufficiently even if the system is sequential. If the system is concurrent or distributed, there can be errors that depend on the order of execution of actions in the system. When a test is executed twice in exactly the same circumstances, such errors may remain undetected or occur only in one of the executions.

Reachability analysis inspects the states of a formal abstract model of the system, aiming to find even such elusive errors. The *complete state space* can be seen as a graph having the states that are reachable from a given initial state of the model as vertices, and all the state transitions between the states as edges. Many properties can easily be checked from such graph if it is finite. If the complete state space is infinite, it is still possible to detect errors by inspecting some finite set of states. On the other hand, a finite complete state space can be far too large with respect to the time and other resources needed to inspect all states in the space. The number of states in the complete state space may grow exponentially or superexponentially with respect to some parameter of the model. We thus have the so called *state space explosion problem.*

Fortunately, many properties can be verified without inspecting all reachable states. For example, reachable terminal states can sometimes be found by inspecting only some of the paths from the initial state to the terminal states. A terminal state can be *acceptable* or *undesirable. Undesirable reachable terminal states* are often called *deadlocks.* Godefroid and Wolper [30] among others have shown how the verification of a *safety property* can often be reduced to the detection of deadlocks. Intuitively, a safety property states what should not happen whereas a *liveness property* states what should happen in the modelled system. A detected error such as a deadlock may be caused by an improper design of the modelled system but it is also possible that the model is improper. The problem whether the model corresponds to the modelled system properly is a challenging area of research. We shall not pursue it further in this thesis, however.

1.2 Two Methods for Efficient Verification

Valmari's *stubborn set method* [70, 71, 73, 74, 75, 76, 78, 79, 80] and Godefroid's *sleep set method* [25, 27, 28, 30, 26, 36, 86, 87] utilize the *independence of state transitions* of the model to eliminate such paths of the complete state space that are redundant with respect to the verification of a given property. These two methods have been combined by Wolper and Godefroid [86], Godefroid and Pirottin [28], and Wolper, Godefroid, and Pirottin [87] to further reduce the number of states inspected during the search. We shall study the stubborn set method, the sleep set method, and their combination in this thesis. We are mainly interested in finding all reachable terminal states by inspecting as few states as possible. All of these methods guarantee that all reachable terminal states are found if the complete state space is finite. It is possible to extend the methods to verify more sophisticated properties, even properties expressed as linear temporal logic formulae [29, 59, 76, 79], but the more properties of the complete state space are preserved, the greater is the number of states inspected during verification. The state space explosion problem often appears even if we limit ourselves to detecting of reachable terminal states. Valmari has shown that the problem whether a reachable terminal state exists is a polynomial space hard problem for finite place/transition nets having a finite set of reachable states [72]. The hardness of the detection of reachable terminal states is not characteristic to concurrent and distributed systems only: the state space explosion problem also appears in sequential systems having nondeterministic choices.

Dijkstra's Dining Philosophers Problem [19] has been widely used in computer science to popularize concurrency control problems. Both the stubborn set method and the sleep set method have been shown to inspect only a polynomial number of states with respect to the number of philosophers in a model of the Dining Philosophers Problem, while the complete state space of the model has an exponential number of states with respect to the number of philosophers [70, 25]. This shows some of the promise contained in the methods.

1.3 Contributions of the Thesis

This work shows that the stubborn set method and the sleep set method can be combined without any of the assumptions previously placed on the stubborn sets as far as the detection of reachable terminal states in place/transition nets is concerned. The obtained result is actually more general and gives a sufficient condition for a method to be compatible with the sleep set method in the detection of reachable terminal states in place/transition nets. We also give justifications for our claim that a result equivalent to the obtained result should hold in the models of concurrency presented by Wolper and Godefroid [86], and Godefroid and Pirottin [28].

The number of enabled transitions in a stubborn set can drastically affect the number of states inspected by the stubborn set method during the search for reachable terminal states. This work presents some heuristics for relieving the problem.

This thesis emphasizes the value of *dynamically stubborn sets* as a useful generalization of stubborn sets and shows some results that improve the understanding of the stubborn set method.

1.4 Related Work

The stubborn set method is closely related to, though not necessarily based on Overman's algorithms [57]. These, according to Valmari [73, 79], are somewhat limited and not so efficient as the stubborn set algorithms. The stubborn set method can also be considered a dynamic priority method in contrary to the static priority method mentioned by Valmari and Tiusanen [82], Rauhamaa [63], and Valmari [79] among others. The static priority method is in turn a generalization of the virtual coarsening of atomic actions presented by Ashcroft and Manna [1], and advocated later by Pnueli [60].

The sleep set method was originally inspired by Mazurkiewicz's *trace theory* [53]. An early version of the sleep set method [25] was essentially faithful to Mazurkiewicz's trace semantics. Later, inspired by Katz's and Peled's work [46], the method has been refined to take into account *conditional independence* [28]. As suggested in [86, 87] and seen in this work, representing traces is sometimes not necessary at all.

Katz and Peled have, independently of Valmari, developed verification algorithms that use *faithful decompositions* [45]. Peled [59] states that faithful decompositions are similar to stubborn sets. Peled has recently improved and extended [59] Valmari's and Godefroid's linear temporal logic verification algorithms.

There is often some *symmetry* in a model. For example, some processes may have been be modelled as if they were executing the same code, which in some communication topologies results in distinct symmetries. It is possible to reduce verification effort by taking into account symmetries and letting an equivalence class of states be represented by one state. Valmari has shown [78] that the stubborn set method is compatible with the symmetry method presented by Huber, Jensen, Jepsen, and Jensen [37] as far as the detection of reachable terminal states is concerned. Later, Tiusanen has shown [69] that the stubborn set method is compatible in a very wide sense with the more general symmetry method presented by Starke [68], and Schmidt and Starke [67]. The symmetry method in [37] includes an algorithm that checks whether two states are equivalent or not. Starke and Schmidt have not presented any corresponding algorithm, but, as mentioned by Clarke, Filkorn, and Jha [12], a unique representative of the equivalence class of a state can be computed using Bryant's BDD's (binary decision diagrams) [10] and Lin's and Newton's method [50, 51]. Emerson and Sistla have also studied symmetries in model checking [20].

During the last few years, much work has been done in the area of *on-the-fly-verification.* This refers to a property being verified during the inspection of states with the inspection being stopped when the truth or falsity of the property is known. Both the stubborn set method and the sleep set method support on-the-fly-verification [29, 80]. Consequently, they are compatible with the memory saving techniques presented by Holzmann [34, 35], Courcoubetis, Vardi, Wolper, and Yannakakis [17], Godefroid, Holzmann, and Pirottin [26], and Wolper and Leroy [88].

There are many approaches attacking the state space explosion problem. We list a few of them, excluding those that have already been mentioned:

- *Time Petri nets* are used as models of *real-time systems.* Earliest and latest firing times are associated with transitions. Yoneda, Shibayama, Schlingloff, and Clarke [89] have presented a method for efficient verification in time Petri nets. They mention that their method shares some ideas with the stubborn set method.

- Janicki and Koutny [38, 39] have presented *optimal simulations* in *state machine decomposable nets.* According to them, these Petri nets have the same expressive power as *1-safe place/transition nets.* Though they do not actually define what is meant by a 1-safe place/transition net, it seems that the definition given later in this thesis is consistent with their concept. We base this claim on the fact that in their state machine decomposable nets, a *self-loop* associated with a transition does not prevent the transition from firing, and also on their description of how a state machine decomposable net can be "collapsed" into a behaviourally equivalent 1-safe place/transition net, and how a 1-safe place/transition net can be transformed into a behaviourally equivalent state machine decomposable net by adding *complement places.* Their optimal simulations respect Mazurkiewicz's trace semantics [38, 39, 53].

- The *occurrence net of a place/transition net* represents the *partial orders of transitions.* McMillan [54] has presented a verification technique that constructs such *fragment of the occurrence net* that is large enough to represent all of the reachable states of the place/transition net. McMillan's technique is oriented to the verification of *asynchronous circuit models.* According to McMillan [54], Valmari's stubborn set method and Godefroid's sleep set method, the latter being called *trace automaton method* in [54], are "ineffective in reducing the state explosion problem for asynchronous circuit models, because of the ubiquity of confusion in such models". The *behaviour machine method* of Probst and Li [61] also builds a representation of the partial orders of transitions and is also oriented to the verification of asynchronous circuit models.

- Reduction rules can be used to transform a Petri net into a smaller net equivalent in some particular sense. Berthelot and Roucairol [3], Berthelot [2], Colom, Martinez, and Silva [15], and Haddad [33] among others have studied net reductions.

- Quemada [62], Dams, Grumberg, and Gerth [18], and Fernandez, Kerbrat, and Mounier [21] among many others have studied *bisimulation.* Here a model is abstracted into a smaller model in such a way that the smaller model and the original model simulate each other in a well-defined sense.

- Burch, Clarke, McMillan, Dill, and Hwang [11] have presented a celebrated *symbolic model checking technique* that expresses many states in one formula of *μ-calculus* and uses a BDD as a normal form.

- It might be possible to develop the *parameterized reachability tree method* of Lindqvist [52] further, though Rauhamaa [63] considers the method impractical and its theoretical profits questionable. A parameterized reachability tree represents many states in one node, parameterized on the values of given *variables.* The basic problem of the method is that the state space represented by the tree may contain states and state transitions that represent no state or state transition in the true state space of the model.

- Vautherin [84] and Findlow [22] employ a more drastic abstraction than Lindqvist. For a high-level net, they construct a *skeleton* by ignoring the identities of tokens. The skeleton can under some circumstances be used to find the reachable terminal states of the original net. Both have presented conditions which guarantee that the set of reachable terminal states of the skeleton corresponds to the set of reachable terminal states of the original net. The skeleton can have a much smaller reachability graph than the original net. However, as shown by Rauhamaa [63], there are instances where the opposite is the case, the skeleton has a larger reachability graph than the original net.

- Murata, Shenker, and Shatz [55] have shown how the linear place and transition invariants of a Petri net can be used in the detection of reachable terminal states. The linear invariants can be obtained from a linear system of equations.

It is sometimes possible to conclude the absence of reachable terminal states from the invariants. More generally, the invariants give candidates that might be reachable terminal states, and one can then perform a restricted, partial state space search to find out whether a candidate is a reachable terminal state or not.

- Compositional verification has been studied by Valmari [77], Graf and Steffen [31], Finkel and Petrucci [23], Valmari and Tienari [81], Kaivola and Valmari [47], and Peled [58] among others. The idea of the compositional verification is that a model is decomposed into modules, the property to be verified is decomposed into subproperties to be verified in the modules, the truth values of which determine the truth value of the property.

To our knowledge, there are only a few computer tools that include the stubborn set method or the sleep set method. Valmari and others have implemented the stubborn set method in the TORAS tool [85], a part of the FORSEE environment [6] of Telecom Australia Research Laboratories as well as in the ARA tool of Technical Research Centre of Finland [48]. Godefroid and others have implemented the sleep set method in the Partial-Order Package [28, 86, 87] which is an add-on package for the SPIN tool of Holzmann [35]. Varpaaniemi has implemented the stubborn set method in the PROD tool developed by Rauhamaa and others in Helsinki University of Technology [32, 83].

1.5 Structure of the Thesis

In Chapter 2, we introduce place/transition nets. The presentation does not go beyond what is necessary for the remaining sections. The stubborn set method is then presented in Chapter 3. Dynamically stubborn sets [63, 78] are shown to be a useful generalization of stubborn sets. Some problems related to the computation of stubborn sets are highlighted. The compatibility of the stubborn set method and high-level nets is also discussed. Section 4 considers the sleep set method and its combination with the stubborn set method. We conclude in Chapter 5 by summarizing the results obtained and briefly discussing possible directions for future research.

2 Place/Transition Nets

In this chapter we give definitions of *place/transition nets* [64, 65] that will be used in later chapters.

The definitions in Section 2.1 are somewhat basic. Section 2.2 is more oriented to the following chapters and introduces the concepts of an *alternative sequence* and the *independence of transitions.*

2.1 Basic Definitions

We shall use "iff" to denote "if and only if". The *power set* (the set of subsets) of a set A is denoted by 2^A. The set of *partial functions* from a set A to a set B is

$$\{R \subseteq A \times B \mid \forall x \in A \; \forall y \in B \; \forall z \in B \; (\langle x, y \rangle \in R \wedge \langle x, z \rangle \in R) \Rightarrow y = z\}.$$

The set of *total relations* from a set A to a set B is

$$\{R \subseteq A \times B \mid \forall x \in A \; \exists y \in B \; \langle x, y \rangle \in R\}.$$

The set of *functions* from a set A to a set B, denoted by $(A \to B)$, is the set of those partial functions from A to B that are total relations from A to B. So there is always an empty function from an empty set to any set but there is no function from a nonempty set to an empty set. The set of natural numbers, including 0, is denoted by N. We shall use ω to denote a formal infinite number, and N_ω to denote $N \cup \{\omega\}$. Relation $\leq$ over N is extended to N_ω by defining

$$\forall n \in N_\omega \; n \leq \omega.$$

Addition and subtraction are extended similarly by defining

$$\forall n \in N \; \omega + n = \omega \wedge \omega - n = \omega.$$

Clearly, $\omega \notin N$ since no natural number can be substituted for ω in these conditions in such a way that the conditions would hold.

Definition 2.1 *A* place/transition net *is a 6-tuple* $\langle S, T, F, K, W, M_0 \rangle$ *such that*

- S *is the set of* places*,*
- T *is the set of* transitions*,* $S \cap T = \emptyset$,
- F *is the set of* arcs*,* $F \subseteq (S \times T) \cup (T \times S)$,
- K *is the* capacity function*,* $K \in (S \to N_\omega)$,

- *W is the* arc weight function*, $W \in (F \to (N \setminus \{0\}))$, and*
- *M_0 is the* initial marking (initial state)*, $M_0 \in \mathcal{M}$ where $\mathcal{M}$ is the set of* markings (states)*, $\mathcal{M} = \{M \in (S \to N) \mid \forall s \in S \; M(s) \leq K(s)\}$.*

If $x \in S \cup T$, then the set of input elements *of x is*

$$^\bullet x = \{y \mid \langle y, x \rangle \in F\},$$

the set of output elements *of x is*

$$x^\bullet = \{y \mid \langle x, y \rangle \in F\},$$

and the set of adjacent elements *of x is $x^\bullet \cup {}^\bullet x$. The function W is extended to a function in $(((S \times T) \cup (T \times S)) \to N)$ by defining $W(x,y) = 0$ iff $\langle x, y \rangle \notin F$. The net is* finite *iff $S \cup T$ is finite. Arcs $\langle x, y \rangle \in F$ and $\langle x', y' \rangle \in F$ form a* self-loop *iff $x' = y$ and $y' = x$. The net has a self-loop iff some arcs of the net form a self-loop. A place s has an* infinite capacity *iff $K(s) = \omega$. The net is a* net with infinite capacities *iff each place has an infinite capacity. For any $k \in N$, the net is k-*safe *iff*

$$\forall s \in S \; \forall t \in T \; K(s) \leq k \wedge W(s,t) \leq k \wedge W(t,s) \leq k.$$

If s is a place of a net and M is a marking of the net, then $M(s)$ is called the number of *tokens* in s at M. If each place has an infinite capacity, then the set of markings of the net is simply $(S \to N)$. Reisig accepts infinite markings, where for some or all $s \in S$, $M(s) = \omega$, in [64] but omits them in [65]. We have omitted infinite markings since infinite markings are redundant in finite place/transition nets and are normally used only as covering markings in coverability graphs [64].

In our figures, places are circles, transitions are rectangles, and the initial marking is shown by the distribution of tokens, black dots, onto places. The weight of an arc is shown iff it is not 1. The capacity of a place is shown iff it is not ω, by the inscription $K = n$.

Definition 2.2 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. A transition t is* enabled at a marking M *iff*

$$\forall s \in {}^\bullet t \; M(s) \geq W(s,t)$$

and

$$\forall s \in t^\bullet \; M(s) - W(s,t) + W(t,s) \leq K(s).$$

A transition t leads (can be fired) from a marking M to a marking M' *($M[t\rangle M'$ for short) iff t is enabled at M and*

$$\forall s \in S \; M'(s) = M(s) - W(s,t) + W(t,s).$$

A transition t is disabled at a marking M *iff t is not enabled at M. A marking M is* terminal *iff no transition is enabled at M. A marking M is* nonterminal *iff M is not terminal. A place s is a* disabling place of a transition t at a marking M *iff $M(s) < W(s,t)$ or $M(s) - W(s,t) + W(t,s) > K(s)$. A partial function f from $\mathcal{M} \times T$ to S is a* scapegoat generator *of the net iff for each marking M and each disabled transition t at M, $f(M,t)$ is a disabling place of t at M.*

If each place has an infinite capacity, then the output places of a transition do not affect the enabledness of the transition. Our enabledness condition is weaker than Reisig's enabledness condition [64, 65] that requires $M(s) + W(t,s) \leq K(s)$ instead of $M(s) - W(s,t) + W(t,s) \leq K(s)$. Our enabledness condition guarantees that our 1-safe place/transition nets are behaviourally equivalent to Godefroid's 1-safe place/transition nets [25]. The definition of a disabling place is consistent with the definition of disabledness since a transition is disabled at a marking iff some place is a disabling place of the transition at the marking. Definition 2.2 does not define a scapegoat generator to be a function since S can be empty.

Finite transition sequences and reachability are introduced in Definition 2.3. We shall use ε to denote the empty sequence.

Definition 2.3 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. For any $T_s \subseteq T$,*

$$\begin{aligned} T_s^0 &= \{\varepsilon\}, \\ (\forall n \in N\ T_s^{n+1} &= \{\sigma t \mid \sigma \in T_s^n \wedge t \in T_s\}),\ \textit{and} \\ T_s^* &= \{\sigma \mid \exists n \in N\ \sigma \in T_s^n\}. \end{aligned}$$

T_s^* is called the set of finite sequences of transitions in T_s, *and T^* is called the* set of finite transition sequences of the net. *A finite transition sequence σ' is a* prefix *of a finite transition sequence σ iff there exists a finite transition sequence σ'' such that $\sigma = \sigma'\sigma''$. A finite transition sequence σ* leads (can be fired) from a marking M to a marking M' *iff $M[\sigma\rangle M'$ where*

$$\forall M \in \mathcal{M}\ M[\varepsilon\rangle M,\ \textit{and}$$

$$\begin{aligned} &\forall M \in \mathcal{M}\ \forall M' \in \mathcal{M}\ \forall \delta \in T^*\ \forall t \in T \\ &M[\delta t\rangle M' \Leftrightarrow (\exists M'' \in \mathcal{M}\ M[\delta\rangle M'' \wedge M''[t\rangle M'). \end{aligned}$$

A finite transition sequence σ is enabled at a marking M *($M[\sigma\rangle$ for short) iff σ leads from M to some marking. A finite transition sequence σ is* disabled at a marking M *iff σ is not enabled at M. A marking M' is* reachable from a marking M *iff some finite transition sequence leads from M to M'. A marking M' is a* reachable marking *iff M' is reachable from M_0. A marking M' is* globally unreachable *iff M' is not reachable from any other marking in $\mathcal{M}$ than M'. For any $k \in N$, the net is* k-bounded *iff for each reachable marking M and for each place $s \in S$, $M(s) \leq k$.*

The (full) reachability graph *of the net is the pair $\langle V, A\rangle$ such that the set of vertices V is the set of reachable markings, and the set of edges A is*

$$\{\langle M, t, M'\rangle \mid M \in V \wedge M' \in V \wedge t \in T \wedge M[t\rangle M'\}.$$

A finite transition sequence is merely a string. It can be thought of as occurring as a path in the full reachability graph iff it is enabled at some reachable marking. Clearly, every k-safe place/transition net is k-bounded, but the converse does not hold. In fact, each place/transition net can be transformed into a behaviourally equivalent net with infinite capacities by adding so called *complement places* [64, 65]. For each k-safe place/transition net, there is thus a behaviourally equivalent k-bounded place/transition net with infinite capacities.

Definition 2.4 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let f be a function from $\mathcal{M}$ to 2^T. A finite transition sequence σ* f-leads (can be f-fired) from a marking M to a marking M' *iff $M[\sigma\rangle_f M'$, where*

$$\forall M \in \mathcal{M}\ M[\varepsilon\rangle_f M, \text{ and}$$

$$\begin{array}{l}\forall M \in \mathcal{M}\ \forall M' \in \mathcal{M}\ \forall \delta \in T^*\ \forall t \in T\\ M[\delta t\rangle_f M' \Leftrightarrow (\exists M'' \in \mathcal{M}\ M[\delta\rangle_f M'' \wedge t \in f(M) \wedge M''[t\rangle M').\end{array}$$

A finite transition sequence σ is f-enabled at a marking M *($M[\sigma\rangle_f$ for short) iff σ f-leads from M to some marking. A marking M' is* f-reachable from a marking *M iff some finite transition sequence f-leads from M to M'. A marking M' is an* f-reachable marking *iff M' is f-reachable from M_0. The* f-reachability graph *of the net is the pair $\langle V, A\rangle$ such that the set of vertices V is the set of f-reachable markings, and the set of edges A is*

$$\{\langle M, t, M'\rangle \mid M \in V \wedge M' \in V \wedge t \in f(M) \wedge M[t\rangle M'\}.$$

Definition 2.4 is like a part of Definition 2.3 except that a transition selection function f determines which transitions are fired. If f is clear from the context or is implicitly assumed to exist and be of a kind that is clear from the context, then the f-reachability graph of the net is called the *reduced reachability graph* of the net. Note that the reduced reachability graph of the net can even be the full reachability graph of the net, e.g. in the case where $f(M) = T$ for each $M \in \mathcal{M}$.

Definition 2.5 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. The* set of infinite transition sequences of the net *is the set of functions from N to T, $(N \to T)$. The function ς from $(N \to T) \times N$ to T^* is defined by*

$$\begin{array}{l}(\forall \sigma \in (N \to T)\ \varsigma(\sigma, 0) = \varepsilon), \quad \text{and}\\ (\forall \sigma \in (N \to T)\ \forall n \in N\ \varsigma(\sigma, n+1) = \varsigma(\sigma, n)\sigma(n)).\end{array}$$

If σ is an infinite transition sequence and $n \in N$, $\varsigma(\sigma, n)$ is called the prefix of length n of σ. *An infinite transition sequence σ is* enabled at a marking M *($M[\sigma\rangle$ for short) iff for each $n \in N$, the prefix of length n of σ is enabled at M. An infinite transition sequence σ is* disabled at a marking M *iff σ is not enabled at M. Let f be a function from $\mathcal{M}$ to 2^T. An infinite transition sequence σ is* f-enabled at a marking M *($M[\sigma\rangle_f$ for short) iff for each $n \in N$, the prefix of length n of σ is f-enabled at M.*

An infinite transition sequence is merely a function. It can be thought of as occurring as a path in the full reachability graph iff it is enabled at some reachable marking.

2.2 Alternative Sequences and the Independence of Transitions

Most of the definitions in Section 2.1 can be found in some form or another in the literature [64, 65]. This section is more oriented to the following chapters. Especially, the concepts of an *alternative sequence* and the *independence of transitions* are introduced.

Definition 2.6 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. A transition sequence δ is an* alternative sequence of a finite transition sequence σ at a marking M *iff δ is a finite transition sequence, σ is enabled at M, and δ leads from M to the same marking as σ. A transition sequence δ is a* length-secure alternative sequence of a finite transition sequence σ at a marking M *iff δ is an alternative sequence of σ at M and not longer than σ. The functions η and ϑ from $T^* \times \mathcal{M}$ to $2^{(T^*)}$ are defined as follows: for each finite transition sequence σ and marking M, $\eta(\sigma, M)$ is the* set of alternative sequences of σ at M, *and $\vartheta(\sigma, M)$ is the* set of length-secure alternative sequences of σ at M.

Clearly, for each finite transition sequence σ and marking M, $\vartheta(\sigma, M) \subseteq \eta(\sigma, M)$. Also, $\eta(\sigma, M)$ is empty iff σ is not enabled at M.

Definition 2.7 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. A transition sequence δ is a* permutation of a finite transition sequence σ *iff δ is a finite transition sequence and for each transition t, the number of t's in δ is equal to the number of t's in σ. A transition sequence δ is an* enabled permutation of a finite transition sequence σ at a marking M *iff δ is a permutation of σ and enabled at M. The function π from $T^* \times \mathcal{M}$ to $2^{(T^*)}$ is defined as follows: for each finite transition sequence σ and marking M, $\pi(\sigma, M)$ is the set of enabled permutations of σ at M.*

Clearly, if finite transition sequences are enabled permutations of each other at a marking M, they lead to the same marking from M. So, if a finite transition sequence σ is enabled at a marking M, then $\pi(\sigma, M) \subseteq \vartheta(\sigma, M)$. The set $\pi(\sigma, M)$ can be nonempty even if σ is not enabled at M since some permutation of σ can be enabled at M. The set of length-secure alternative sequences, as well as the set of alternative sequences, of an enabled finite transition sequence σ at a marking can always be partitioned into sets of enabled permutations of sequences at the marking. Of course, only one of those sets is the set of enabled permutations of σ.

Definition 2.8 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Transitions t and t'* commute at a marking *M iff $M[tt'\rangle$ and $M[t't\rangle$. Transitions t and t' are* independent at a marking *M iff*

$$(M[tt'\rangle \wedge M[t't\rangle) \vee (\neg M[t\rangle \wedge \neg M[t'\rangle) \vee$$
$$(M[t\rangle \wedge \neg M[t'\rangle \wedge \neg M[tt'\rangle) \vee (M[t'\rangle \wedge \neg M[t\rangle \wedge \neg M[t't\rangle).$$

Our definition of independence corresponds to Godefroid's and Pirottin's [28] definition of conditional independence which in turn is based on Katz's and Peled's [46] corresponding definition. Our definition of independence can be obtained from Godefroid's and Pirottin's definition of valid conditional dependency relations, Definition 5 in [28], by taking the necessary conditions for a triple of two transitions and one state to be in the complement of a valid dependency relation, and substituting terms of place/transition nets for the terms of the model of concurrency in [28] in an obvious way.

The following can clearly be seen from the above.

- Different transitions are independent at a marking iff neither of them can be fired at the marking making the other transition turn from enabled to disabled or from disabled to enabled.
- A transition t commutes with itself at a marking iff tt is enabled at the marking.
- A transition t is independent of itself at a marking iff tt is enabled or t is disabled at the marking.
- Transitions commute at a marking iff they are enabled and independent at the marking.

Definition 2.9 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. A transition sequence δ is a* neighbour *of a finite transition sequence σ iff there exist transitions t and t', and finite transition sequences σ' and σ'' such that $\sigma = \sigma' t t' \sigma''$ and $\delta = \sigma' t' t \sigma''$. A transition sequence δ is an* enabled neighbour *of a finite transition sequence σ at a marking M iff δ is a neighbour of σ and enabled at M. Let M be a marking and R*

$\eta(\sigma, M)$	the set of alternative sequences of a finite transition sequence σ at M
$\vartheta(\sigma, M)$	the set of length-secure alternative sequences of a finite transition sequence σ at M
$\pi(\sigma, M)$	the set of enabled permutations of a finite transition sequence σ at M
$\iota(\sigma, M)$	the conditional trace of a finite transition sequence σ at M

Figure 1: The functions η, ϑ, π, and ι.

the binary relation on T^* such that $\sigma R \delta$ iff σ and δ are enabled neighbours of each other at M. The conditional trace of a finite transition sequence σ at M *is the set of finite transition sequences such that a sequence δ is in the conditional trace of σ at M iff σ is enabled at M and $\sigma R^* \delta$ where R^* is the reflexive-transitive closure of R. A set is a* conditional trace at M *iff the set is the conditional trace of some finite transition sequence at M. The function ι from $T^* \times \mathcal{M}$ to $2^{(T^*)}$ is defined as follows: for each finite transition sequence σ, and marking M, $\iota(\sigma, M)$ is the conditional trace of σ at M.*

In other words, a conditional trace is a set of enabled finite transition sequences at a marking that can be obtained from each other by repeatedly interchanging adjacent independent transitions. We did not have to mention independence in Definition 2.9 since transitions commute at a marking iff they are enabled and independent at the marking. The reflexive-transitive closure of R in Definition 2.9 is clearly an equivalence relation, and the conditional trace of a finite transition sequence is the equivalence class of the sequence with respect to the equivalence relation. Our definition of a conditional trace corresponds to Godefroid's and Pirottin's [28] definition which in turn is based on Katz's and Peled's [46] corresponding definition. The conditional trace of an enabled finite transition sequence at a marking is naturally a subset of the enabled permutations of the sequence at the marking. Thus $\iota(\sigma, M) \subseteq \pi(\sigma, M)$ holds for each σ and M. Moreover, the set of enabled permutations of an enabled finite transition sequence at a marking can always be partitioned into conditional traces at the marking. Of course, only one of those conditional traces is the conditional trace of the sequence. Note that if a finite transition sequence σ is disabled at a marking M, then the conditional trace of σ at M is empty.

Figure 1 presents the functions η, ϑ, π, and ι in a nutshell.

A conditional trace is a generalization of a Mazurkiewicz's trace [53]. Mazurkiewicz presented his traces for *condition/event nets*. A condition/event net is behaviourally close to a 1-safe place/transition net that does not have a self-loop. However, the concept of a Mazurkiewicz's trace can meaningfully be generalized to concern all place/transition nets. Assuming the convention used by Wolper and Godefroid [86], we can say that a Mazurkiewicz's trace is a set of finite transition sequences that can be obtained from each other by repeatedly interchanging adjacent globally independent transitions. There are many levels of global independence. We could define that transitions are globally independent iff they are independent at every marking that is reachable from the initial marking [28]. Or, we could define that transitions t and t' are globally independent iff no adjacent place of t is an adjacent place of t' [25, 53]. The former definition has the problem that it is computationally hard to check whether two transitions are globally independent. The latter definition is somewhat coarse if the net has self-loops.

Definition 2.10 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let f be a function from $\mathcal{M}$ to 2^T. Then we say that f* represents all sets of alternative sequences to terminal markings *iff*

$$\forall \sigma \in T^* \ \forall M \in \mathcal{M} \ (M[\sigma\rangle \wedge \forall t \in T \ \neg M[\sigma t\rangle) \Rightarrow (\exists \delta \in \eta(\sigma, M) \ \ M[\delta\rangle_f).$$

Correspondingly, f represents all sets of length-secure alternative sequences to terminal markings *iff*

$$\forall \sigma \in T^* \ \forall M \in \mathcal{M} \ (M[\sigma\rangle \wedge \forall t \in T \ \neg M[\sigma t\rangle) \Rightarrow (\exists \delta \in \vartheta(\sigma, M) \ \ M[\delta\rangle_f).$$

Respectively, f represents all sets of enabled permutations to terminal markings *iff*

$$\forall \sigma \in T^* \ \forall M \in \mathcal{M} \ (M[\sigma\rangle \wedge \forall t \in T \ \neg M[\sigma t\rangle) \Rightarrow (\exists \delta \in \pi(\sigma, M) \ \ M[\delta\rangle_f).$$

Finally, f represents all conditional traces to terminal markings *iff*

$$\forall \sigma \in T^* \ \forall M \in \mathcal{M} \ (M[\sigma\rangle \wedge \forall t \in T \ \neg M[\sigma t\rangle) \Rightarrow (\exists \delta \in \iota(\sigma, M) \ \ M[\delta\rangle_f).$$

The following can clearly be seen from the above.

- A function repersenting all conditional traces to terminal markings represents all sets of enabled permutations to terminal markings.
- A function representing all sets of enabled permutations to terminal markings represents all sets of length-secure alternative sequences to terminal markings.
- A function representing all sets of length-secure alternative sequences to terminal markings represents all sets of alternative sequences to terminal markings.

3 Stubborn Set Method

In this chapter we present Valmari's *stubborn set method* [70, 71, 73, 74, 75, 76, 78, 79, 80]. Section 3.1 is concentrated on *dynamically stubborn* sets [63, 78]. All the stubborn sets that have been defined in the literature are known to be dynamically stubborn. Dynamically stubborn sets seem to have all the nice properties of (statically) stubborn sets except that the definition of dynamic stubbornness does not seem to imply a practical algorithm for computing dynamically stubborn sets. We base the definitions on Rauhamaa's principles [63]. We also present Godefroid's and Pirottin's definitions of *persistent* and *conditionally stubborn* sets [28] in the context of place/transition nets.

In Section 3.2, we show that dynamically stubborn sets are really useful. We consider how reachable terminal markings are preserved by a *dynamically stubborn set selective reachability graph generation*, that is, reachability graph generation that at each encountered marking selects a dynamically stubborn set and fires the enabled transitions in the set, without firing other transitions. All reachable terminal markings are shown to occur in the reduced reachability graph. Also, if there is an infinite path in the full reachability graph, then the reduced reachability graph is shown to have an infinite path, too. We end Section 3.2 by showing a property which is the basis for Valmari's algorithms for detecting *ignored transitions* and eliminating the *ignoring phenomenon* [75]. A transition is ignored at a marking iff the transition is enabled at the marking but not fired at any marking that is reachable from the marking. The existence of ignored transitions is called the ignoring phenomenon.

True (or *static*) *stubbornness* is defined in Section 3.3. As we are interested in detecting reachable terminal markings by inspecting as few markings as possible, we have chosen a very weak definition of stubbornness. Our definition is the definition in [70] modified by taking advantage of the remarks in [70].

The *incremental algorithm* for computing stubborn sets [70, 73] is presented in Section 3.4, and the *deletion algorithm* [71, 73] in Section 3.5. The deletion algorithm is slower than the incremental algorithm, but the incremental algorithm is not guaranteed to produce minimal stubborn sets in any practical sense, unlike the deletion algorithm. The compatibility of the stubborn set method and high-level nets [8, 43] is discussed in Section 3.6.

One critical factor in the incremental algorithm possibly producing unnecessarily large stubborn sets is the choice of a *scapegoat*. A scapegoat is a disabling place chosen during the execution of the incremental algorithm for a disabled transition. In Section 3.7, the gravity of the problem together with some heuristics for choosing a scapegoat are described in the context of a classical example.

3.1 Dynamic Stubbornness

We define dynamic stubbornness on the basis of Rauhamaa's principles [63].

Definition 3.1 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let M be a marking of the net. A set $T_s \subseteq T$* fulfils the first principle of dynamic stubbornness *(*D1 *for short)* at M *iff*

$$\forall \sigma \in (T \setminus T_s)^* \ \forall t \in T_s \ M[\sigma t\rangle \Rightarrow M[t\sigma\rangle.$$

A transition t is a key transition of a set $T_s \subseteq T$ at M *iff $t \in T_s$ and*

$$\forall \sigma \in (T \setminus T_s)^* \ M[\sigma\rangle \Rightarrow M[\sigma t\rangle.$$

A set $T_s \subseteq T$ fulfils the second principle of dynamic stubbornness *(*D2 *for short)* at M *iff T_s has a key transition at M. A set $T_s \subseteq T$* fulfils the principle of conventional dynamic stubbornness *(*CD *for short)* at M *iff*

$$\forall \sigma \in (T \setminus T_s)^* \ \forall \delta \in (T \setminus T_s)^* \ \forall t \in T_s \ M[\sigma \delta t\rangle \Rightarrow M[\sigma t \delta\rangle.$$

A set $T_s \subseteq T$ fulfils the first principle of strong dynamic stubbornness *(*SD1 *for short)* at M *iff*

$$\forall \sigma \in (T \setminus T_s)^* \ \forall t \in T_s \ M[\sigma t\rangle \Rightarrow M[t\rangle.$$

A set $T_s \subseteq T$ fulfils the second principle of strong dynamic stubbornness *(*SD2 *for short)* at M *iff*

$$\forall \sigma \in (T \setminus T_s)^* \ \forall t \in T_s \ (M[t\rangle \wedge M[\sigma\rangle) \Rightarrow (M[\sigma t\rangle \wedge M[t\sigma\rangle).$$

A set $T_s \subseteq T$ is dynamically stubborn at M *iff T_s fulfils D1 and D2 at M. A set $T_s \subseteq T$ is* conventionally dynamically stubborn at M *iff T_s fulfils CD and D2 at M. A set $T_s \subseteq T$ is* unconventionally dynamically stubborn at M *iff T_s is dynamically stubborn but not conventionally dynamically stubborn at M. A set $T_s \subseteq T$ is* strongly dynamically stubborn at M *iff T_s fulfils SD1 and SD2 at M and $\exists t \in T_s \ M[t\rangle$.*

The principles D1, D2, CD, SD1, and SD2 are illustrated in Figure 2. The principles D1, D2, SD1, and SD2 are Rauhamaa's Principles 1*, 2*, 1, and 2, respectively [63]. Clearly, a key transition of a set at a marking is enabled at the marking. Our key transitions are similar to Valmari's key transitions [75]. The difference is that Valmari's key transitions satisfy a condition that can be checked easily and is sufficient but not necessary for a transition to be a key transition in the sense of our definition.

We shall see in Section 3.2 that dynamic stubbornness alone is sufficient as far as the detection of reachable terminal markings is concerned, conventional dynamic

$$\begin{tikzcd}
M \arrow[r, "\sigma"] & M1 \arrow[d, "t"'] & D1 & M \arrow[r, "\sigma"] \arrow[d, "t"'] & M1 \arrow[d, "t"'] \\
 & M2 & \text{=>} & M3 \arrow[r, "\sigma"] & M2
\end{tikzcd}$$

$$\begin{tikzcd}
M \arrow[r, "\sigma"] & M1 & D2 & M \arrow[r, "\sigma"] & M1 \arrow[d, "t\ (key)"'] \\
 & & \text{=>} & & M2
\end{tikzcd}$$

$$\begin{tikzcd}
M \arrow[d, "\sigma"'] & & & M \arrow[d, "\sigma"'] & \\
M1 \arrow[d, "\delta"'] & & CD\ \text{=>} & M1 \arrow[r, "t"] \arrow[d, "\delta"'] & M4 \arrow[d, "\delta"'] \\
M2 \arrow[r, "t"] & M3 & & M2 \arrow[r, "t"] & M3
\end{tikzcd}$$

$$\begin{tikzcd}
M \arrow[r, "\sigma"] & M1 \arrow[d, "t"'] & SD1 & M \arrow[r, "\sigma"] \arrow[d, "t"'] & M1 \arrow[d, "t"'] \\
 & M2 & \text{=>} & M3 & M2
\end{tikzcd}$$

$$\begin{tikzcd}
M \arrow[r, "\sigma"] \arrow[d, "t"'] & M2 & SD2 & M \arrow[r, "\sigma"] \arrow[d, "t"'] & M2 \arrow[d, "t"'] \\
M1 & & \text{=>} & M1 \arrow[r, "\sigma"] & M3
\end{tikzcd}$$

Figure 2: The principles of dynamic, conventional dynamic, and strong dynamic stubbornness.

stubbornness is related to conditional traces, and strongly dynamically stubborn sets are useful when one wants to eliminate the ignoring phenomenon. The term "conventional" refers to the often used heuristic that every sequence of such transitions that are not in a given stubborn set should leave the set stubborn. Valmari has shown that the heuristic has some advantages when the so called *candidate list algorithm*, briefly discussed in Section 3.3, is used for computing stubborn sets [70].

Valmari has also defined dynamically stubborn sets [78]. Valmari's dynamically stubborn sets are considered later in this section.

Lemma 3.2 *If a set is conventionally dynamically stubborn at a marking, the set is dynamically stubborn set at the marking. A set is strongly dynamically stubborn at a marking iff the set is dynamically stubborn at the marking and each enabled transition in the set is a key transition of the set at the marking.*

Proof. The results follow trivially from Definition 3.1. □

The result in Lemma 3.2 is due to Valmari [75] but has missed explicit treatment.

As one might expect, unconventionally dynamically stubborn sets exist. In the net in Figure 3, $\{a, b\}$ is dynamically stubborn but not conventionally dynamically stubborn at the initial marking since $M_0[cdeb\rangle$ and $\neg M_0[cdb\rangle$. The set $\{c\}$ is strongly dynamically stubborn at the initial marking.

A set can be conventionally dynamically stubborn without being strongly dynamically stubborn. In the net in Figure 4, the only dynamically stubborn sets at the initial marking are $\{t_0, t_1\}$, $\{t_1, t_2\}$, and $\{t_0, t_1, t_2\}$. The sets $\{t_0, t_1\}$ and $\{t_1, t_2\}$ are conventionally dynamically stubborn but not strongly dynamically stubborn at the initial marking since $\neg M_0[t_1 t_2\rangle$ and $\neg M_0[t_1 t_0\rangle$.

Lemma 3.3 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let M be a marking of the net. A set $T_s \subseteq T$ fulfils SD2 at M iff*

$$\forall \sigma \in (T \setminus T_s)^* \ \forall \delta \in (T \setminus T_s)^* \ \forall t \in T_s \ (M[t\rangle \wedge M[\sigma\delta\rangle) \Rightarrow M[\sigma t \delta\rangle.$$

Proof. The "if"-part is obvious. Let's prove the "only if" -part. Let a set $T_s \subseteq T$ fulfil SD2 at M. Let $\sigma \in (T \setminus T_s)^*$, $\delta \in (T \setminus T_s)^*$, $t \in T_s$, $M[t\rangle$, and $M[\sigma\delta\rangle$. Using SD2 for both $\sigma\delta$ and σ, we get $M[t\sigma\delta\rangle$ and $M[\sigma t\rangle$. As σt and $t\sigma$ lead to the same marking, we have $M[\sigma t \delta\rangle$. □

The result in Lemma 3.3 is due to Valmari [75] but has missed explicit treatment.

Lemma 3.4 *If a set is strongly dynamically stubborn at a marking, the set is conventionally dynamically stubborn at the marking.*

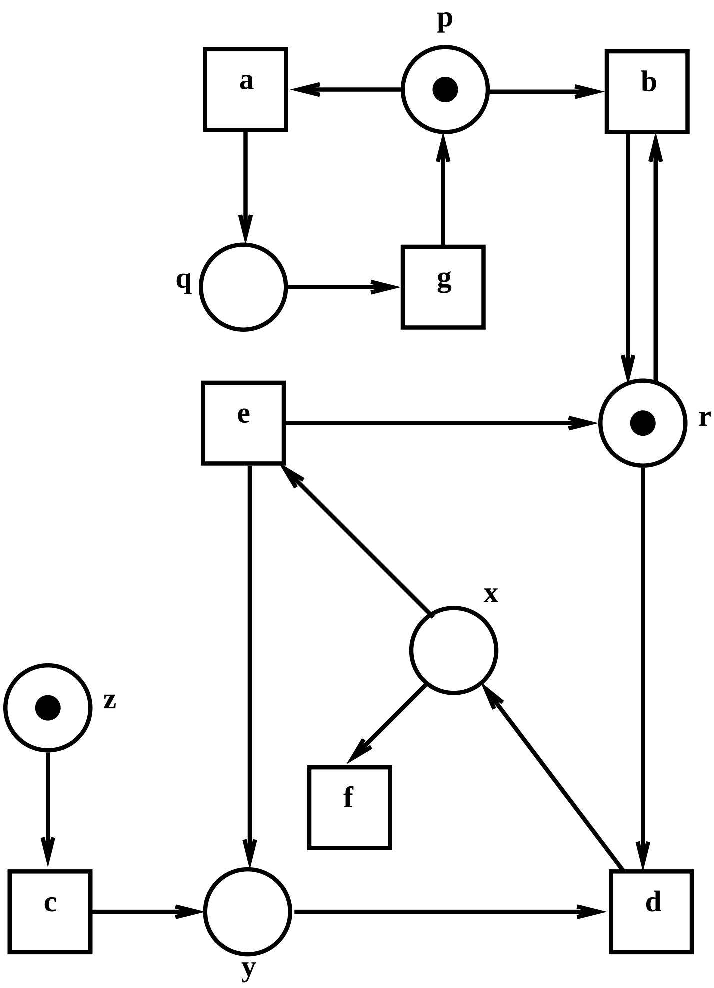


Figure 3: The set $\{a, b\}$ is unconventionally dynamically stubborn.

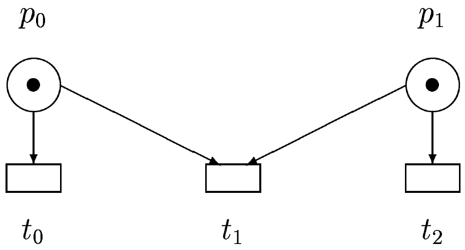


Figure 4: The set $\{t_0, t_1\}$ is conventionally but not strongly dynamically stubborn.

Proof. The result follows trivially from Definition 3.1 and Lemma 3.3. □

The result in Lemma 3.4 is due to Valmari [75] but has missed explicit treatment.

Lemma 3.5 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let $M \in \mathcal{M}$. A set $T_s \subseteq T$ is strongly dynamically stubborn at M iff*

$$\forall \sigma \in (T \setminus T_s)^* \ \forall M' \in \mathcal{M}$$
$$M[\sigma\rangle M' \Rightarrow (T_s \textit{ is strongly dynamically stubborn at } M').$$

Proof. The "if"-part is obvious since $M[\varepsilon\rangle M$. Let's prove the "only if" -part. Let a set $T_s \subseteq T$ be strongly dynamically stubborn at M. Let $\sigma \in (T \setminus T_s)^*$, $\delta \in (T \setminus T_s)^*$, $t \in T_s$, $M' \in \mathcal{M}$, and $M[\sigma\rangle M'$. If $M'[\delta t\rangle$, we have $M[\sigma\delta t\rangle$, so by SD1 and SD2 for M it follows that $M[\sigma t\rangle$, which implies $M'[t\rangle$. The set T_s thus fulfils SD1 at M'. If $M'[t\rangle$ and $M'[\delta\rangle$, we have $M[\sigma t\rangle$ and $M[\sigma\delta\rangle$, so by SD1 and SD2 for M and Lemma 3.3 it follows that $M[\sigma\delta t\rangle$ and $M[\sigma t\delta\rangle$, which implies $M'[\delta t\rangle$ and $M'[t\delta\rangle$. The set T_s thus fulfils SD2 at M'. If $M[t\rangle$, SD2 for M implies $M[\sigma t\rangle$, so $M'[t\rangle$. Some transition in T_s is thus enabled at M' since some transition in T_s is enabled at M. □

The result in Lemma 3.5 is new though inspired by Valmari [75].

Lemma 3.5 states that every sequence of such transitions that are not in a given strongly dynamically stubborn set leaves the set strongly dynamically stubborn. Lemma 3.6 states the similar result for conventionally dynamically stubborn sets.

Lemma 3.6 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let $M \in \mathcal{M}$. A set $T_s \subseteq T$ is conventionally dynamically stubborn at M iff*

$$\forall \sigma \in (T \setminus T_s)^* \ \forall M' \in \mathcal{M}$$
$$M[\sigma\rangle M' \Rightarrow (T_s \textit{ is conventionally dynamically stubborn at } M').$$

Proof. The "if"-part is obvious since $M[\varepsilon\rangle M$. Let's prove the "only if" -part. Let a set $T_s \subseteq T$ be conventionally dynamically stubborn at M. Let $\sigma \in (T \setminus T_s)^*$, $\delta \in (T \setminus T_s)^*$, $\delta' \in (T \setminus T_s)^*$, $t \in T_s$, $M' \in \mathcal{M}$, and $M[\sigma\rangle M'$. If $M'[\delta\delta' t\rangle$, we have $M[\sigma\delta\delta' t\rangle$, so by CD for M it follows that $M[\sigma\delta t\delta'\rangle$, which implies $M'[\delta t\delta'\rangle$. The set T_s thus fulfils CD at M'. Let τ be a key transition of T_s at M. If $M'[\delta\rangle$, we have $M[\sigma\delta\rangle$, so by D2 for M it follows that $M[\sigma\delta\tau\rangle$. The transition τ is thus a key transition of T_s at M'. □

The result in Lemma 3.6 is new though inspired by Valmari [75].

There is no lemma analogous to Lemmas 3.5 and 3.6 for all dynamically stubborn sets. Let $M_0[c\rangle M'$ and $M'[d\rangle M''$ in the net in Figure 3. The set $\{a, b\}$ is dynamically stubborn at M_0 and M' but not at M'' since $M''[eb\rangle$ and $\neg M''[b\rangle$.

Lemma 3.7 states that a set is conventionally dynamically stubborn iff the set is dynamically stubborn and every sequence of such transitions that are not in the set leaves the set dynamically stubborn.

Lemma 3.7 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let $M \in \mathcal{M}$. A set $T_s \subseteq T$ is conventionally dynamically stubborn at M iff*

$$\begin{aligned}&\forall \sigma \in (T \setminus T_s)^* \ \forall M' \in \mathcal{M} \\ &M[\sigma\rangle M' \Rightarrow (T_s \textit{ is dynamically stubborn at } M').\end{aligned}$$

Proof. The "only if" -part follows directly from Lemmas 3.2 and 3.6. Let's prove the "if"-part. Let $T_s \subseteq T$, and

$$\forall \sigma' \in (T \setminus T_s)^* \ \forall M' \in \mathcal{M} \ M[\sigma'\rangle M' \Rightarrow (T_s \text{ is dynamically stubborn at } M').$$

Let $\sigma \in (T \setminus T_s)$, $\delta \in (T \setminus T_s)$, $t \in T_s$, and $M' \in \mathcal{M}$ be such that $M[\sigma\delta t\rangle$ and $M[\sigma\rangle M'$. By D1 for M' we have $M'[t\delta\rangle$, so $M[\sigma t\delta\rangle$. The set T_s thus fulfils CD at M. Since $M[\varepsilon\rangle M$, T_s is dynamically stubborn at M. The set T_s thus fulfils D2 at M. □

The result in Lemma 3.7 is new though inspired by Valmari [70].

Lemma 3.7 gives a useful alternative characterization of conventionally dynamically stubborn sets. We shall use this alternative characterization in Section 3.3.

Lemma 3.8 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let M be a marking of the net, and T_s and T_e subsets of T such that*

$$\{t \in T_s \mid M[t\rangle\} \subseteq T_e, \ \textit{and } T_e \subseteq T_s.$$

If T_s is dynamically stubborn at M, T_e is dynamically stubborn at M. If T_s is conventionally dynamically stubborn at M, T_e is conventionally dynamically stubborn at M. If T_s is strongly dynamically stubborn at M, T_e is strongly dynamically stubborn at M.

Proof. (i) Let T_s be dynamically stubborn at M. We show that

$$\forall \sigma \in (T \setminus T_e)^* \ M[\sigma\rangle \Rightarrow \sigma \in (T \setminus T_s)^*.$$

Let $\sigma \in (T \setminus T_e)^*$ and $\delta \in (T \setminus T_s)^*$ be such that $M[\sigma\rangle$ and δ is the longest prefix of σ not containing any transition in T_s. If $\delta \neq \sigma$, the first transition after δ in σ is enabled at M by D1 for T_s. Since no transition in $T_s \setminus T_e$ is enabled at M, we conclude that $\delta = \sigma$, so $\sigma \in (T \setminus T_s)^*$.

(ii) Since conventionally dynamically stubborn sets and strongly dynamically stubborn sets are dynamically stubborn by Lemma 3.2, the result of part (i) holds for them, too. Then D1 for T_s implies D1 for T_e, D2 for T_s implies D2 for T_e, CD for T_s, implies CD for T_e, SD1 for T_s implies SD1 for T_e, and SD2 for T_s, implies SD2 for T_e. □

The result in Lemma 3.8 is new though inspired by Godefroid and Pirottin [28].

Lemma 3.8 states that if we remove disabled transitions from a dynamically stubborn (conventionally dynamically stubborn, strongly dynamically stubborn) set, the remaining set is dynamically stubborn (conventionally dynamically stubborn, strongly dynamically stubborn). For example, if a dynamically stubborn set is minimal with respect to set inclusion, by Lemma 3.8 the set consists of enabled transitions only.

We define persistence and conditional stubbornness in such a way that the definitions correspond to the definitions given by Godefroid and Pirottin [28]. Our definitions can be obtained from Godefroid's and Pirottin's Definitions 7 and 8 in [28] by substituting terms of place/transition nets for the terms of the model of concurrency in [28] in an obvious way.

Definition 3.9 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let $M \in \mathcal{M}$. A set $T_s \subseteq T$* fulfils the principle of persistence and conditional stubbornness *(*PE *for short)* at *M iff*

$$\begin{array}{l}\forall \sigma \in (T \setminus T_s)^* \ \forall t \in T_s \ \forall t' \in T \setminus T_s \ \forall M' \in \mathcal{M} \\ (M[t\rangle \wedge M[\sigma\rangle M' \wedge M'[t'\rangle) \Rightarrow \\ (t \textit{ and } t' \textit{ are independent at } M').\end{array}$$

A set $T_s \subseteq T$ is persistent at *M iff T_s fulfils PE at M and $\forall t \in T_s \ M[t\rangle$. A set $T_s \subseteq T$ is* conditionally stubborn at *M iff T_s fulfils SD1 and PE at M and $\exists t \in T_s \ M[t\rangle$.*

Clearly, the "$M[t\rangle \wedge$" in PE is redundant in the definition of persistence since all transitions in persistent sets are enabled. Looking at PE and proceeding inductively with respect to the length of σ, one observes that "commute" could be substituted for "are independent" in PE. Combining this observation with Lemma 3.3, one concludes that PE is nothing but SD2. Consequently, we rid ourselves of the concepts of persistence and conditional stubbornness

Lemma 3.10 *A set fulfils PE at a marking iff the set fulfils SD2 at the marking. A set is conditionally stubborn at a marking iff the set is strongly dynamically stubborn at the marking.*

Proof. We show that PE is equivalent to SD2. The second statement then follows directly from Definitions 3.1 and 3.9. Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let $M \in \mathcal{M}$ and $T_s \subseteq T$.

(i) We prove that SD2 implies PE. Let T_s fulfil SD2 at M. Let $\sigma \in (T \setminus T_s)^*$, $t \in T_s$, $t' \in T \setminus T_s$, $M' \in \mathcal{M}$, $M[t\rangle$, $M[\sigma\rangle M'$, and $M'[t'\rangle$. By Lemma 3.3 we have both $M'[t't\rangle$ and $M'[tt'\rangle$. The transitions t and t' are thus independent at M'.

(ii) We prove that PE implies SD2. Let T_s fulfil PE at M. We use induction on the length of finite transition sequences to show that T_s fulfils SD2 at M. The principle

SD2 is fulfilled trivially when restricted to ε. Our induction hypothesis is that SD2 is fulfilled when restricted to finite transition sequences of length $n \geq 0$. We show that SD2 is then fulfilled when restricted to finite transition sequences of length $n+1$. Let $\delta \in (T \setminus T_s)^*$, $t' \in T \setminus T_s$, and $t \in T_s$ be such that $M[t\rangle$, $M[\delta t'\rangle$, and δ is of length n. Let $M' \in \mathcal{M}$ be such that $M[\delta\rangle M'$. The transition t' is then enabled at M'. By the induction hypothesis we have $M[\delta t\rangle$ and $M[t\delta\rangle$. The transition t is thus enabled at M'. The principle PE then implies that t and t' are independent at M'. Transitions commute at a marking iff they are enabled and independent at the marking. So t and t' commute at M'. Thus $M'[tt'\rangle$ and $M'[t't\rangle$, and consequently $M[\delta tt'\rangle$ and $M[\delta t't\rangle$. As already mentioned, we have $M[t\delta\rangle$, so $t\delta$ leads from M to the same marking as δt. We thus have $M[t\delta t'\rangle$. □

The result in Lemma 3.10 is new.

Lemma 3.11 *A set is a nonempty persistent set at a marking iff the set is a conditionally stubborn set at the marking and does not contain any transition that is disabled at the marking. The set of enabled transitions of any conditionally stubborn set is a nonempty persistent set.*

Proof. The first statement follows from the fact that a persistent set fulfils SD1 trivially since all its transitions are enabled. The second statement follows trivially from the first statement and Lemmas 3.8 and 3.10. □

The result in Lemma 3.11 is due to Godefroid and Pirottin [28] but the proof is new.

Lemma 3.12 *A set is a nonempty persistent set at a marking iff the set is a strongly dynamically stubborn set at the marking and does not contain any transition that is disabled at the marking. The set of enabled transitions of any strongly dynamically stubborn set is a nonempty persistent set.*

Proof. The result follows trivially from Lemmas 3.10 and 3.11. □

We now turn to Valmari's dynamically stubborn sets [78]. The prefix AV used in the sequel comes from the name Antti Valmari.

Definition 3.13 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let M be a marking of the net. A set $T_s \subseteq T$* fulfils the principle of AV-strong dynamic stubbornness *(*AVSD *for short)* at M *iff*

$$\forall t \in T_s\ \forall t' \in T \setminus T_s\ \forall M' \in \mathcal{M}\ \ (M[t\rangle \wedge M'[t\rangle \wedge M'[t'\rangle) \Rightarrow (M[tt'\rangle \wedge M[t't\rangle).$$

A set $T_s \subseteq T$ is AV-strongly dynamically stubborn at M *iff T_s fulfils SD1 and AVSD at M and $\exists t \in T_s\ M[t\rangle$.*

$$\begin{tikzcd}
M \arrow[d, "t"'] & M' \arrow[r, "t'"] \arrow[d, "t"'] & M3 & \overset{AVSD}{=>} & M' \arrow[r, "t'"] \arrow[d, "t"'] & M3 \arrow[d, "t"'] \\
M1 & M2 & & & M2 \arrow[r, "t'"] & M4
\end{tikzcd}$$

Figure 5: The principle of AV-strong dynamic stubbornness.

Our AV-strong dynamic stubbornness is equivalent to Valmari's strong dynamic stubbornness [78], because of the obvious equivalence between our Definition 3.13 and Valmari's Definition 2.2 in [78]. The principle AVSD is illustrated in Figure 5.

Lemma 3.14 *If a set is AV-strongly dynamically stubborn set at a marking, the set is strongly dynamically stubborn at the marking.*

Proof. Valmari's Theorem 2.5 in [78] shows that if a set is AV-strongly dynamically stubborn set at a marking, the set fulfils D1 at the marking. An AV-strongly dynamically stubborn set contains an enabled transition by Definition 3.13. By Lemma 3.2 it then suffices to show that each enabled transition of an AV-strongly dynamically stubborn set at a marking is a key transition of the set at the marking. Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let M be a marking of the net, and T_s a subset of T such that T_s is AV-strongly dynamically stubborn at M. Let a transition $t \in T_s$ be enabled at M. We show that

$$\forall \sigma \in (T \setminus T_s)^* \; M[\sigma\rangle \Rightarrow M[\sigma t\rangle.$$

We use induction on the length of σ. The claim holds trivially when restricted to $\sigma = \varepsilon$. Our induction hypothesis is that the claim holds when restricted to any σ of length $n \geq 0$. Let $\sigma \in (T \setminus T_s)^*$, $t' \in T \setminus T_s$, and $M' \in \mathcal{M}$ be such that σ is of length n, $M[\sigma\rangle M'$ and $M'[t'\rangle$. Using the induction hypothesis, we get $M[\sigma t\rangle$ which implies $M'[t\rangle$. We already have $M[t\rangle$ and $M'[t'\rangle$. Since T_s fulfils AVSD at M, $M'[t't\rangle$. Thus $M[\sigma t' t\rangle$. □

The result in Lemma 3.10 is due to Valmari [78] but has missed explicit treatment.

The converse of Lemma 3.14 does not hold. In the net in Figure 6, $\{c\}$ is strongly dynamically stubborn but not AV-strongly dynamically stubborn at the initial marking since $M_0[cc\rangle$, $M_0[cd\rangle$, and $\neg M_0[ccd\rangle$.

The motivation behind the definition of AV-strongly dynamically stubborn sets seems to be related to the aim to find static conditions for computing statically stubborn sets. We shall discuss this subject in Section 3.4.

Figure 7 illustrates the classes of dynamically, conventionally dynamically, strongly dynamically, and AV-strongly dynamically stubborn sets at a marking M. The in-

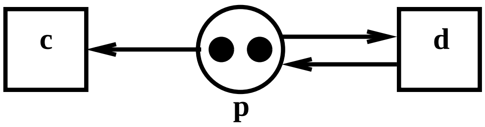


Figure 6: The set $\{c\}$ is strongly but not AV-strongly dynamically stubborn.

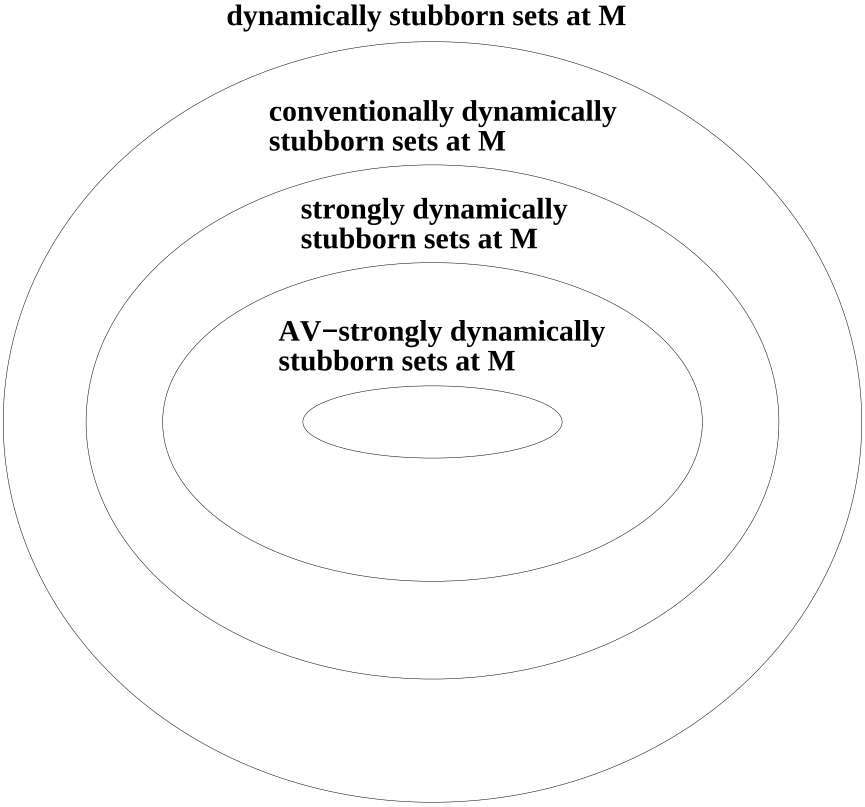

dynamically stubborn sets at M
conventionally dynamically stubborn sets at M
strongly dynamically stubborn sets at M
AV−strongly dynamically stubborn sets at M

Figure 7: Four classes of dynamically stubborn sets at a marking.

clusions follow from Lemmas 3.2, 3.4, and 3.14. By the presented examples related to Figures 3, 4, and 6 we know that the inclusions are strict.

3.2 Usefulness of Dynamically Stubborn Sets

We now consider what is preserved by a dynamically stubborn set selective reachability graph generation. The results presented in this section show the usefulness of dynamically stubborn sets.

Definition 3.15 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net. Let f be a function from $\mathcal{M}$ to 2^T. Then we say that f is* dynamically stubborn *iff for each nonterminal marking M, $f(M)$ is dynamically stubborn. Correspondingly, f is* conventionally dynamically stubborn *iff for each nonterminal marking M, $f(M)$ is conventionally dynamically stubborn. Respectively, f is* unconventionally dynamically stubborn *iff f is dynamically stubborn but not conventionally dynamically stubborn. Correspondingly, f is* strongly dynamically stubborn *iff for each nonterminal marking M, $f(M)$ is strongly dynamically stubborn. Finally, f is* AV-strongly dynamically stubborn *iff for each nonterminal marking M, $f(M)$ is AV-strongly dynamically stubborn.*

Theorem 3.16 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let f be a dynamically stubborn function from $\mathcal{M}$ to 2^T. Then f represents all sets of enabled permutations to terminal markings.*

Proof. We show that

$$\forall \sigma'' \in T^* \ \forall M \in \mathcal{M} \ (M[\sigma''\rangle \land \forall t \in T \ \neg M[\sigma'' t\rangle) \Rightarrow (\exists \delta \in \pi(\sigma'', M) \ M[\delta\rangle_f).$$

We use induction on the length of σ''. The claim holds trivially when restricted to $\sigma'' = \varepsilon$. Our induction hypothesis is that the claim holds when restricted to any σ'' of length $n \geq 0$. Let $\sigma \in T^*$, $M \in \mathcal{M}$, and $M' \in \mathcal{M}$ be such that σ is of length $n+1$, $M[\sigma\rangle M'$, and $\forall t \in T \ \neg M'[t\rangle$. The set $f(M)$ is dynamically stubborn at M since some transition is enabled at M. The sequence σ must contain a transition in $f(M)$ since otherwise some transition in $f(M)$ would be enabled at M' by D2. Let $\delta \in (T \backslash f(M))^*$, $t \in f(M)$, and $\delta' \in T^*$ be such that $\sigma = \delta t \delta'$. By D1 we have $M[t\delta\rangle$, so $M[t\delta\delta'\rangle$. Let $M'' \in \mathcal{M}$ be such that $M[t\rangle M''$. Now $M[t\rangle_f M''$. By the induction hypothesis, $\exists \sigma' \in \pi(\delta\delta', M'') \ M''[\sigma'\rangle_f$. We thus have $t\sigma' \in \pi(\sigma, M)$ and $M[t\sigma'\rangle_f$. □

The result in Theorem 3.16 is due to Valmari [70, 75] but has missed explicit treatment.

Theorem 3.16 has the consequence that if a finite transition sequence leads from a marking M to a terminal marking, and M occurs in the reduced reachability graph, then an enabled permutation of the sequence occurs in the graph. A dynamically stubborn set selective search thus certainly finds all reachable terminal markings if the net and the set of reachable markings are finite. If the set of reachable markings is infinite but the net is finite and a dynamically stubborn set selective search is performed in a breadth-first order for some time, then reachable terminal markings "near the initial marking" can be found. As we shall see in Chapter 4, the permutation preserving property makes the stubborn set method compatible with the sleep set method in the detection of reachable terminal markings though a weaker property would suffice.

Theorem 3.17 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let f be a conventionally dynamically stubborn function from $\mathcal{M}$ to 2^T. Then f represents all conditional traces to terminal markings.*

Proof. We show that

$$\forall \sigma'' \in T^* \; \forall M \in \mathcal{M} \; (M[\sigma''\rangle \wedge \forall t \in T \; \neg M[\sigma'' t\rangle) \Rightarrow (\exists \delta \in \iota(\sigma'', M) \; M[\delta\rangle_f).$$

We use induction on the length of σ''. The claim holds trivially when restricted to $\sigma'' = \varepsilon$. Our induction hypothesis is that the claim holds when restricted to any σ'' of length $n \geq 0$. Let $\sigma \in T^*$, $M \in \mathcal{M}$, and $M' \in \mathcal{M}$ be such that σ is of length $n+1$, $M[\sigma\rangle M'$, and $\forall t \in T \; \neg M'[t\rangle$. The set $f(M)$ is conventionally dynamically stubborn at M since some transition is enabled at M. The sequence σ must contain a transition in $f(M)$ since otherwise some transition in $f(M)$ would be enabled at M' by D2. Let $\delta \in (T \setminus f(M))^*$, $t \in f(M)$, and $\delta' \in T^*$ be such that $\sigma = \delta t \delta'$. By CD we have $M[t\delta\rangle$, so $M[t\delta\delta'\rangle$. Moreover, CD implies that $t\delta$ is in the conditional trace of δt at M. The sequence $t\delta\delta'$ is thus in $\iota(\sigma, M)$, the conditional trace of σ at M. Let $M'' \in \mathcal{M}$ be such that $M[t\rangle M''$. Now $M[t\rangle_f M''$. By the induction hypothesis, $\exists \sigma' \in \iota(\delta\delta', M'') \; M''[\sigma'\rangle_f$. We thus have $t\sigma' \in \iota(\sigma, M)$ and $M[t\sigma'\rangle_f$. □

The result in Theorem 3.17 is new though inspired by Wolper and Godefroid [86].

Theorem 3.16 has the consequence that if f is a conventionally dynamically stubborn function, a finite transition sequence leads from a marking M to a terminal marking, and M occurs in the f-reachability graph, then some member of the conditional trace of the sequence occurs in the graph.

There are dynamically stubborn functions that do not represent all conditional traces to terminal markings. Let h be any dynamically stubborn function in the net in Figure 3 such that $h(M_0) = \{a, b\}$. The transition sequence *cdebdf* leads from M_0 to a terminal marking. The function h does not represent the conditional trace of *cdebdf* at M_0 since *cdebdf* is the only member of the conditional trace of *cdebdf* at M_0.

There are unconventionally dynamically stubborn functions that represent all conditional traces to terminal markings. Let's consider an example having the additional property that some set is dynamically stubborn but not conventionally dynamically stubborn at some marking from which some terminal marking is reachable. In the net in Figure 8, $\{a, b\}$ is dynamically stubborn but not conventionally dynamically stubborn at the initial marking since $M_0[cdb\rangle$ and $\neg M_0[cb\rangle$. Let f be the unconventionally dynamically stubborn function defined by $f(M_0) = \{a, b\}$ and $f(M) = T$ when $M \neq M_0$. There is one and only one terminal marking that is reachable from M_0. Let M' be the terminal marking and $M_0[\sigma\rangle M'$. Since a is independent of other transitions at all markings, there exists δ such that $a\delta$ is in the conditional trace of σ at M_0. For each $M \in \mathcal{M}$, no transition leads from M to M_0, so M_0 is globally unreachable by Definition 2.3. The global unreachability of M_0 and the definition of f now imply $M_0[a\delta\rangle_f$. If $M \neq M_0$ and $M[\sigma'\rangle$, the global unreachability of M_0 and the definition of f imply $M[\sigma'\rangle_f$. We have thus shown that f represents all conditional traces to terminal markings.

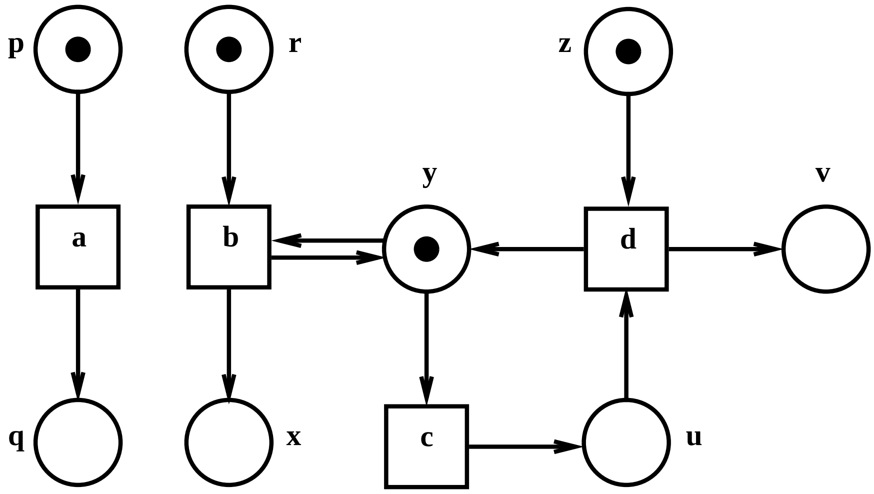


Figure 8: An unconventionally dynamically stubborn f represents all conditional traces to terminal markings.

Theorem 3.18 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let f be a dynamically stubborn function from $\mathcal{M}$ to 2^T. Then for each marking M and for each infinite transition sequence σ, if σ is enabled at M, there exists an infinite transition sequence σ' such that $M[\sigma'\rangle_f$.*

Proof. Let an infinite transition sequence σ be enabled at a marking M. We construct an infinite transition sequence σ' and a function δ from N to the set of infinite transition sequences such that for each n and k in N, $M[\varsigma(\sigma', n)\rangle_f$ and $M[\varsigma(\sigma', n)\varsigma(\delta(n), k)\rangle$.

We use induction on n. We set $\delta(0) = \sigma$. Our induction hypothesis is that $\sigma'(k)$ has been set for each $k < n$ and $\delta(k)$ for each $k \leq n$ in such a way that $M[\varsigma(\sigma', n)\rangle_f$ and $\forall k \in N\ M[\varsigma(\sigma', n)\varsigma(\delta(n), k)\rangle$. Our goal is to set $\sigma'(n)$ and $\delta(n+1)$ in such a way that $M[\varsigma(\sigma', n+1)\rangle_f$ and $\forall k \in N\ M[\varsigma(\sigma', n+1)\varsigma(\delta(n+1), k)\rangle$. Let $M' \in \mathcal{M}$ be such that $M[\varsigma(\sigma', n)\rangle_f M'$. The set $f(M')$ is dynamically stubborn at M' since $M'[\varsigma(\delta(n), 1)\rangle$.

Let's first consider the case where $\exists j \in N\ \delta(n)(j) \in f(M')$. Let i be the least such j. Let $M'' \in \mathcal{M}$ be such that $M'[\varsigma(\delta(n), i{+}1)\rangle M''$. Then $\varsigma(\delta(n), i{+}1) = \varsigma(\delta(n), i)\delta(n)(i)$ by Definition 2.5. By D1 we have $M'[\delta(n)(i)\varsigma(\delta(n), i)\rangle M''$. We reach our goal by setting $\sigma'(n) = \delta(n)(i)$ and

$$\forall k \in N\ (k < i \wedge \delta(n+1)(k) = \delta(n)(k)) \vee (k \geq i \wedge \delta(n+1)(k) = \delta(n)(k+1)).$$

Let's then consider the case where $\forall k \in N\ \delta(n)(k) \notin f(M')$. By D2, a key transition of $f(M')$ at M' exists. Let t'' be any key transition of $f(M')$ at M'. Then $\forall k \in N\ M[\varsigma(\delta(n), k)t''\rangle$. Using D1 we get $\forall k \in N\ M[t''\varsigma(\delta(n), k)\rangle$. We reach our goal by setting $\sigma'(n) = t''$ and $\delta(n+1) = \delta(n)$. □

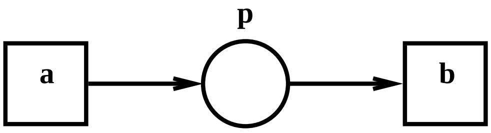


Figure 9: Finite reduced reachability graph though the full reachability graph is infinite.

The result in Theorem 3.18 is due to Valmari [74, 75, 78] but has missed explicit treatment. Our proof is like the proof in [78] which is more general than the proof in [75].

Theorem 3.18 states that for each marking in the reduced reachability graph, if an infinite transition sequence is enabled at the marking, the graph contains an infinite path that starts from the marking. If there is no loop in a finite reduced reachability graph, Theorem 3.18 implies that the full reachability graph is finite and has no loop either. If one instead generates a finite reduced reachability graph having a loop, it is still possible that the full reachability graph is infinite. Let $M_0[a\rangle M'$ in the net in Figure 9. Let's define f by $f(M_0) = \{a\}$, $f(M') = \{b\}$, and $f(M) = T$ for other markings M. Then f is an AV-strongly dynamically stubborn function, and the f-reachability graph has only two vertices and two edges though the full reachability graph is infinite.

We now turn to the *ignoring phenomenon*. A transition is ignored at a marking iff the transition is enabled at the marking but not fired at any marking that is reachable from the marking [75]. The existence of ignored transitions is called the ignoring phenomenon. For example, if there is an isolated transition, we easily get a reduced reachability graph containing only one vertex and one edge, thus leaving the behaviour of the other parts of the net uninvestigated. Valmari has shown [75] that if ignoring does not occur, a transition occurs in the reduced reachability graph iff it occurs in the full reachability graph. Liveness of a transition in the sense defined by Reisig [64] is also preserved if ignoring does not occur [75].

Valmari has developed an algorithm for detecting ignored transitions and an algorithm for eliminating the ignoring phenomenon. The elimination algorithm requires that the chosen dynamically stubborn sets are strongly dynamically stubborn. The detection algorithm does not have that limitation. The detection algorithm works in time at most linear in the number of vertices and edges of the reduced reachability graph. The elimination algorithm works in time at most proportional to the number of vertices and edges of the resulting reduced reachability graph multiplied by the number of transitions of the net [75].

Valmari's algorithms for detecting and eliminating the ignoring phenomenon are based on the property that is stated in Lemma 3.19.

Lemma 3.19 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net. Let f and g be functions from $\mathcal{M}$ to 2^T such that f is dynamically stubborn and*

$$\forall M \in \mathcal{M}\ g(M) \subseteq \{t \in f(M) \mid \forall \delta \in (T \setminus f(M))^*\ M[\delta\rangle \Rightarrow M[\delta t\rangle\}.$$

Then

$$\forall \sigma \in T^*\ \forall t \in T\ \forall M \in \mathcal{M}\ (M[\sigma\rangle_g \wedge M[t\rangle \wedge \forall \delta \in T^*\ \neg M[\delta t\rangle_f) \Rightarrow M[\sigma t\rangle.$$

Proof. We use induction on the length of the above σ. The above claim holds trivially when restricted to an empty σ. Our induction hypothesis is that the above claim holds when restricted to every σ of length $n \geq 0$. Let $\sigma \in T^*$, $t' \in T$, $t \in T$, $M \in \mathcal{M}$, and $M' \in \mathcal{M}$ be such that σ is of length n, $M[t'\rangle_g M'$, $M'[\sigma\rangle_g$, $M[t\rangle$, and $\forall \delta \in T^*\ \neg M[\delta t\rangle_f$. The set $f(M)$ is dynamically stubborn at M since $M[t\rangle$. As $M[t\rangle$ and $\neg M[t\rangle_f$, we have $t \notin f(M)$. As $t' \in g(M)$, we then have $t' \in f(M)$ and $M[tt'\rangle$. Using D1 we get $M[t't\rangle$, so $M'[t\rangle$. From $M[t'\rangle_f M'$ and $\forall \delta \in T^*\ \neg M[\delta t\rangle_f$ it follows that $\forall \delta' \in T^*\ \neg M'[\delta' t\rangle_f$. Using the induction hypothesis we get $M'[\sigma t\rangle$, so $M[t'\sigma t\rangle$. □

The result in Lemma 3.19 is due to Valmari [75] but has missed explicit treatment.

Let's assume that f is a dynamically stubborn function, and there is a path leading from a marking M to a marking M' in the f-reachability graph such that for each edge $\langle M'', t, M'''\rangle$ on the path, t is a key transition of $f(M'')$ at M''. From Lemma 3.19 it then follows that any transition ignored at M is ignored at M', too.

Let f and g be as in Lemma 3.19. By Definition 3.1,

$$\{t \in f(M) \mid \forall \delta \in (T \setminus T_s)^*\ M[\delta\rangle \Rightarrow M[\delta t\rangle\}$$

is then the set of key transitions of $f(M)$ at M and nonempty. We can thus require that $g(M)$ is nonempty when M is nonterminal. Let's further assume that the g-reachability graph is finite. For each transition that is ignored at some marking in the f-reachability graph, by Lemma 3.19 there is then a terminal maximal strongly connected component of the g-reachability graph such that for each marking in the component, the transition is enabled but not fired at the marking. Valmari's algorithm for detecting ignored transitions inspects the terminal maximal strongly connected components of such g-reachability graph [75].

All enabled transitions of a strongly dynamically stubborn set are key transitions of the set by Lemma 3.2. If f is a strongly dynamically stubborn function, we can then choose $g(M)$ to be the set of enabled transitions in $f(M)$ with the consequence that the g-reachability graph is the f-reachability graph. Let's further assume that the f-reachability graph is finite. For each transition that is ignored at some marking in the f-reachability graph, there is then a terminal maximal strongly connected component of the f-reachability graph such that for each marking in the component, the transition is enabled but not fired at the marking. Valmari's algorithm for eliminating the ignoring phenomenon utilizes this property [75].

3.3 Stubbornness

We define true stubbornness in place/transition nets with infinite capacities. As mentioned in the discussion after Definition 2.3, each place/transition net can be transformed into a behaviourally equivalent net with infinite capacities by adding complement places [64, 65]. As we are interested in detecting reachable terminal markings by inspecting as few markings as possible, we have chosen a very weak definition of stubbornness.

Definition 3.20 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net with infinite capacities. The function E_1 from $\mathcal{M} \times S$ to 2^T, the functions E_2 and E_3 from $\mathcal{M} \times T \times S$ to 2^T, and the function E_4 from S to 2^T are defined as follows: let $M \in \mathcal{M}$, $t \in T$, and $s \in S$. Then*

$$\begin{aligned}
E_1(M, s) &= \{t' \in {}^\bullet s \mid M(s) \geq W(s, t') \wedge W(t', s) > W(s, t')\}, \\
E_2(M, t, s) &= E_4(s) \cup \{t' \in s^\bullet \mid \; W(s, t) > W(t, s) \wedge \\
&\qquad\qquad\qquad\qquad\quad W(s, t') > M(s) - W(s, t) + W(t, s)\}, \\
E_3(M, t, s) &= E_1(M, s) \cup \{t' \in {}^\bullet s \mid M(s) \geq W(s, t') \wedge W(t', s) > W(t, s)\}, \textit{ and} \\
E_4(s) &= \{t' \in s^\bullet \mid W(s, t') > W(t', s)\}.
\end{aligned}$$

Intuitively, $E_1(M, s)$ is the set of transitions that could increase the number of tokens in s and are not disabled by s at M. Correspondingly, $E_2(M, t, s)$ is the set of transitions that could decrease the number of tokens in s or get disabled because of the firing of t at M. Respectively, $E_3(M, t, s)$ is the set of transitions that are not disabled by s at M and could increase the number of tokens in s or deposit more tokens to s than t. Finally, $E_4(s)$ is the set of transitions that could decrease the number of tokens in s.

Definition 3.21 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net with infinite capacities. A set $T_s \subseteq T$ is* stubborn at a marking M *iff*

$$\begin{aligned}
\forall t \in T_s \;\; & (\exists s \in {}^\bullet t \; M(s) < W(s, t) \wedge E_1(M, s) \subseteq T_s) \vee \\
& (M[t\rangle \wedge (\forall s \in {}^\bullet t \; W(s, t) \leq W(t, s) \vee E_2(M, t, s) \subseteq T_s \vee E_3(M, t, s) \subseteq T_s))
\end{aligned}$$

and

$$\exists \tau \in T_s \; M[\tau\rangle \wedge (\forall s \in {}^\bullet \tau \; E_4(s) \subseteq T_s).$$

Definition 3.21 is the definition in [70] modified by taking advantage of the remarks in [70]. According to Valmari [75], the definition in [70] is not intuitive. We use Definition 3.21 merely because it gives us a feasible way to compute dynamically stubborn sets and is very weak. The weakness is good as far as the detection of reachable terminal markings is concerned, since the weaker is the definition of

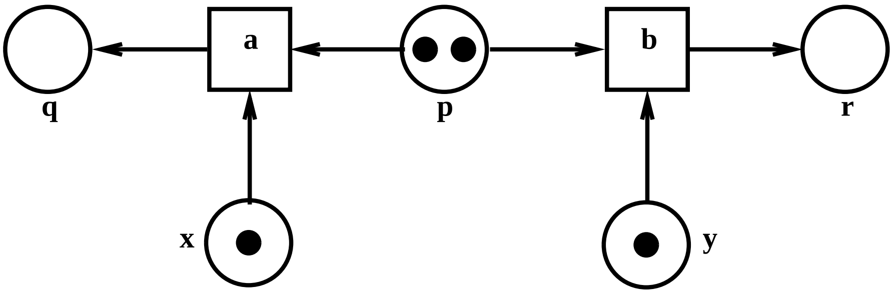


Figure 10: The set $\{a\}$ is strongly dynamically stubborn but not stubborn.

stubbornness, the better are the chances to find stubborn sets having a small number of enabled transitions. We do not know any better simple heuristic for minimizing the number of markings inspected during the search for reachable terminal markings than the minimization of the number of transitions fired at a marking.

Theorem 3.22 *If a set is stubborn at a marking, the set is dynamically stubborn at the marking.*

Proof. The definition of E_4 guarantees that any transition matching the τ in Definition 3.21 is a key transition of T_s at M. Valmari has shown in the proof of his Theorem 2.2 and in the remarks immediately below the proof in [70] that our stubborn sets fulfil D1. □

The result in Theorem 3.22 is due to Valmari [70].

If we remove "$W(s,t) \leq W(t,s) \vee$" from Definition 3.21, we get a definition for the stubborn sets in [70]. Such stubborn set is conventionally dynamically stubborn since by Valmari's Lemma 2.5 in [70], every sequence of such transitions that are not in the set leaves the set stubborn, and we can then use our Theorem 3.22 and Lemma 3.7.

Let's return to some of the examples of Section 3.1. In the net in Figure 3, $\{a,b\}$ is stubborn but not conventionally dynamically stubborn, and $\{c\}$ is stubborn and strongly dynamically stubborn at the initial marking. In the net in Figure 4, $\{t_0,t_1\}$ and $\{t_1,t_2\}$ are stubborn and conventionally dynamically stubborn but not strongly dynamically stubborn at the initial marking. In the net in Figure 6, $\{c\}$ is stubborn and strongly dynamically stubborn but not AV-strongly stubborn at the initial marking.

As one might expect, a dynamically stubborn set is not necessarily stubborn. In the net in Figure 10, $\{a\}$ and $\{b\}$ are strongly dynamically stubborn but not stubborn at the initial marking.

Valmari has presented three algorithms for finding a suitable stubborn set: the *candidate list algorithm* [70], the *incremental algorithm* [70, 73], and the *deletion* *algorithm* [71, 73]. Let's assume that T is the set of transitions and μ is the maximum number of input places of a transition. The candidate list algorithm selects the first stubborn set in a given candidate list T_1, ..., T_n, T. The time taken by an execution of the candidate list algorithm is at most proportional to $\mu \sum_{i=1}^{n} |T_i|$. The candidate list determines the size of the reduced reachability graph. We do not know good heuristics for automatic candidate list construction, so we concentrate on the incremental algorithm and the deletion algorithm that are automatic by nature. Both of these two algorithms contain nondeterministic choices, and there are cases where manual preliminary preparations can be useful.

Lemma 3.10 suggests a natural definition of strong dynamic stubbornness in the models of concurrency of [28, 86]: a set is strongly dynamically stubborn iff it is conditionally stubborn. Godefroid and Pirottin [28], and Wolper and Godefroid [86] have defined static stubbornness corresponding to conditional stubbornness and shown how statically stubborn sets can be computed much in the same way as in place/transition nets, with the limitation that the computed sets are conditionally stubborn. On the other hand, we do not currently know how to define other levels of dynamic stubbornness in the models of concurrency of [28, 86] in a useful way. There are two essential difficulties. The first difficulty is the fact that in those models, it matters in which order the transitions are fired: there can be finite transition sequences σ and δ such that σ and δ are enabled permutations of each other at a state M but lead to different states from M. The second difficulty is the problem of how to compute dynamically stubborn sets: a definition of dynamic stubbornness should be justified by a corresponding definition of static stubbornness and a feasible algorithm to compute statically stubborn sets.

Godefroid and Pirottin [28] have presented a technique for finding some conditionally stubborn sets that, according to Godefroid and Pirottin [28], cannot be easily found by algorithms that are based on static definitions of stubbornness. The technique seems to require a lot of human work, but Godefroid and Pirottin have presented cases where the technique seems to be justified [28].

3.4 Incremental Algorithm

We now turn to the incremental algorithm [70, 73] for computing stubborn sets.

Definition 3.23 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a place/transition net with infinite capacities. Let G be the set of scapegoat generators of the net and B the set of functions from $\mathcal{M} \times T \times S$ to $\{2, 3\}$. The function E_{14} from $G \times B \times \mathcal{M} \times T$ to 2^T is defined by*

$$\forall f \in G\ \forall b \in B\ \forall M \in \mathcal{M}\ \forall t \in T$$

$$E_{14}(f, b, M, t) = \begin{cases} E_1(M, f(M, t)) & \textit{if } \neg M[t\rangle, \\ \bigcup_{s \in {}^\bullet t}(E_{b(M,t,s)}(M, t, s) \cup E_4(s)) & \textit{if } M[t\rangle. \end{cases}$$

The function R_{14} *from* $G \times B \times \mathcal{M}$ *to* $2^{T \times T}$ *is defined by*

$$\forall f \in G \; \forall b \in B \; \forall M \in \mathcal{M}$$
$$R_{14}(f, b, M) = \{\langle t, t' \rangle \in T \times T \mid t' \in E_{14}(f, b, M, t)\}.$$

For each $f \in G$, $b \in B$, *and* $M \in \mathcal{M}$, *the reflexive-transitive closure of* $R_{14}(f, b, M)$ *is denoted by* $(R_{14}(f, b, M))^*$. *The function* E_{14}^* *from* $G \times B \times \mathcal{M} \times T$ *to* 2^T *is defined by*

$$\forall f \in G \; \forall b \in B \; \forall M \in \mathcal{M} \; \forall t \in T$$
$$E_{14}^*(f, b, M, t) = \{t' \mid \langle t, t' \rangle \in (R_{14}(f, b, M))^*\}.$$

For each $f \in G$, $b \in B$, *and* $M \in \mathcal{M}$, *the* $\langle f, b \rangle$-dependency graph at M *is the pair* $\langle V, A \rangle$ *such that the set of vertices* V *is* T, *and the set of edges* A *is* $R_{14}(f, b, M)$.

Clearly, the set $E_{14}(f, b, M, t)$ is the set of transitions immediately succeeding t in the $\langle f, b \rangle$-dependency graph at M. Respectively, $E_{14}^*(f, b, M, t)$ is the set of transitions accessible from t in the $\langle f, b \rangle$-dependency graph at M. Note that Definition 3.20 implies $E_2(M, t, s) \cup E_4(s) = E_2(M, t, s)$.

Lemma 3.24 *Let* $\langle S, T, F, K, W, M_0 \rangle$ *be a place/transition net with infinite capacities. Let* G *be the set of scapegoat generators of the net and* B *the set of functions from* $\mathcal{M} \times T \times S$ *to* $\{2, 3\}$. *Let* $f \in G$, $b \in B$, $M \in \mathcal{M}$, $t \in T$, *and* $M[t\rangle$. *Then* $E_{14}^*(f, b, M, t)$ *is both stubborn and strongly dynamically stubborn at* M.

Proof. Stubbornness is obvious. Theorem 3.22 then implies dynamic stubbornness. Strong dynamic stubbornness follows from dynamic stubbornness and the definition of E_4. □

The result in Lemma 3.24 is due to Valmari [70, 75].

The stubborn set in Lemma 3.24 is stubborn in the sense of the definition in [70], since the "$W(s, t) \leq W(t, s) \vee$" in Definition 3.21 is not utilized in Definition 3.23.

The incremental algorithm in [70] modified for our definitions can be described as follows. Let $\langle S, T, F, K, W, M_0 \rangle$ be a finite place/transition net with infinite capacities. Let G be the set of scapegoat generators of the net and B the set of functions from $\mathcal{M} \times T \times S$ to $\{2, 3\}$. Let $f \in G$, $b \in B$, and $M \in \mathcal{M}$ be such that M is nonterminal. The algorithm produces a set T_s such that for some enabled transition τ at M, $T_s = E_{14}^*(f, b, M, \tau)$, and $\forall t \in T_s \; M[t\rangle \Rightarrow \tau \in E_{14}^*(f, b, M, t)$. The enabled transitions of T_s are in one maximal strongly connected component of the $\langle f, b \rangle$-dependency graph at M. The enabled transitions of T_s are found by traversing the $\langle f, b \rangle$-dependency graph in depth-first order, starting from an enabled transition, applying Tarjan's algorithm for computing maximal strongly connected components [66], and stopping when the first maximal strongly connected component having an enabled transition is found.

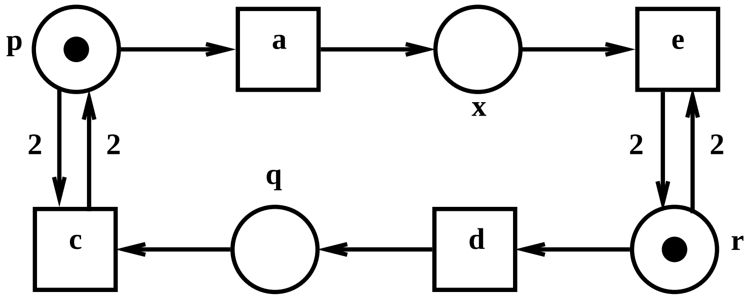


Figure 11: The function E_3 is potentially useful in the incremental algorithm.

The time taken by an execution of this algorithm is at most proportional to $\mu\nu|T|$, where μ is the maximum number of input places of a transition, and ν is the maximum number of adjacent transitions of a place. The number μ can be $|S|$, the number ν can be $|T|$, and $|S|$ can be far greater than $|T|$. We have made the practical assumption that the time taken by the computation of $f(M,t)$ is at most proportional to $\mu\nu$, and the time taken by the computation of $b(M,t,s)$ is at most proportional to ν. Even if we assumed that the computation of f and b takes no time, we would have the same time complexity for the incremental algorithm.

Without change in complexity, the incremental algorithm can be optimized [79, 83]: T_s is chosen to be such $E^*_{14}(f,b,M,\tau)$ that contains the least number of enabled transitions. (As above, τ has to be enabled at M.) All what is needed is to complete the depth-first search and application of Tarjan's algorithm in such a way that all enabled and only enabled transitions are checked in the outermost loop of the search. If the optimized incremental algorithm is used, the functions f and b are the only nondeterministic factors that affect the number of enabled transitions in T_s.

The reachability graph generation algorithm using the incremental algorithm produces a g-reachability graph of the net such that for each marking M in the graph, $g(M)$ is the T_s at M if there is an enabled transition at M. (If no transition is enabled at M, then any subset of T is valid for $g(M)$.)

The function b in Definition 3.23 makes our incremental algorithm more refined than the incremental algorithm in [70]. If b has the value 2 everywhere, our algorithm behaves as the algorithm in [70]. Let's consider the net in Figure 11. If $b(M_0,a,p) = b(M_0,d,r) = 3$, the computed stubborn set at M_0 is either $\{a\}$ or $\{d\}$, independently of the scapegoat generator. Let f be a scapegoat generator such that $f(M_0,c) = q$ and $f(M_0,e) = x$. If f is used in the computation and $b(M_0,a,p) = b(M_0,d,r) = 2$, the computed stubborn set at M_0 is $\{a,c,d,e\}$.

The refinement is due to Valmari [70] but has missed explicit treatment. The incremental algorithm in [70] is a "compromise between generality and practicality" [70], and so is our incremental algorithm since our algorithm computes strongly dynamically stubborn sets though we are mainly interested in the detection of reachable terminal markings.

The incremental algorithm sometimes produces stubborn sets that are not AV-strongly dynamically stubborn, even if the choice function b has the value 2 everywhere. Let's consider the net in Figure 6. If $b(M_0, c, p) = 2$, the computed stubborn set at M_0 is $\{c\}$, independently of the scapegoat generator. By the remark immediately after Lemma 3.24 we know that $\{c\}$ is stubborn also in the sense of the definition in [70]. The set $\{c\}$ is strongly dynamically stubborn but not AV-strongly dynamically stubborn at M_0 since $M_0[cc\rangle$, $M_0[cd\rangle$, and $\neg M_0[ccd\rangle$.

Of course, we can choose another definition of static stubbornness if we want to compute AV-strongly dynamically stubborn sets only. Such definitions occur in [78]. The "unnecessary strength" of AV-strong dynamic stubbornness is caused by the global condition in AVSD. The global condition seems to be related to Valmari's older comment: "... stubborn set theory aims at finding a sufficient and statically computable condition ..." [75]. Definition 3.21 contains many conditions that depend on the marking and are thus not statically computable.

We end this section by describing what is meant by a pseudo-random scapegoat generator. An explicit description of a pseudo-random scapegoat generator would be far too complicated to be presented here. Therefore, only an informal description is given. Let $r_1, r_2, r_3, \ldots$ be an infinite pseudo-random number sequence [66]. Let f be the scapegoat generator to be defined. For each marking M and transition t, $f(M, t)$ is not defined earlier than necessary. If $\langle M, t\rangle$ is the ith pair for which $f(M, t)$ has to be defined, then $f(M, t)$ is defined to be the $((r_i \bmod k) + 1)$th disabling place of t at M, where the list of disabling places of t at M is of length k and determined by some fixed list of the input places of t.

3.5 Deletion Algorithm

The stubborn sets computed by the incremental algorithm may contain unnecessarily many enabled transitions. To solve this problem, Valmari has developed the deletion algorithm [71, 73]. The deletion algorithm finds a stubborn set which is minimal in the sense that no proper subset of its enabled transitions can be the set of enabled transitions of any stubborn set. The deletion algorithm utilizes the definition of stubbornness completely, unlike the incremental algorithm. To our knowledge, no one has presented any algorithm that would find a stubborn set having a minimum number of enabled transitions in polynomial time with respect to the number of places and transitions. On the other hand, Valmari has given an example [73] where always choosing such set leads unavoidably to a greater number of vertices and a greater number of edges in the reduced reachability graph than there are in a reduced reachability graph produced by a certain different strategy. Of course, if we are to choose between a set of enabled transitions and its proper subset, it is better to choose the proper subset.

Looking at the deletion algorithm for so called variable/transition systems in [71], it is not quite obvious how to get a deletion algorithm for place/transition nets. Though variable/transition systems can be transformed into place/transition nets, the problem remains that the definition of stubbornness in [71] is much stronger than our definition. Moreover, both the incremental and the deletion algorithm utilize the whole definition of stubbornness in [71], which may make one draw incorrect conclusions about the correspondence between the incremental and the deletion algorithm in general. A solution to our problem, given by Valmari in a private discussion, is to derive the deletion algorithm from the definition of stubbornness in a way faithful to the definition of stubbornness. This general idea is used in Definition 3.25.

Definition 3.25 *Let $\langle S, T, F, K, W, M_0\rangle$ be a place/transition net with infinite capacities and M a marking of the net. The* and/or-graph at M *is a triple $\langle V_{AND}, V_{OR}, A\rangle$ such that the set of and-vertices V_{AND} is*

$$\begin{array}{l} \{s \mid \exists t \in T\ \neg M[t\rangle \wedge s \in {}^\bullet t\} \cup \\ \{t \in T \mid M[t\rangle\} \cup \\ \{\langle t, s, i\rangle \mid t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s) \wedge i \in \{2,3\}\}, \end{array}$$

the set of or-vertices V_{OR} is

$$\begin{array}{l} \{t \in T \mid \neg M[t\rangle\} \cup \\ \{\langle t, s\rangle \mid t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s)\}, \end{array}$$

and the set of edges A is

$$\begin{array}{ll} \{\langle s, t'\rangle \mid \exists t \in T\ \neg M[t\rangle \wedge s \in {}^\bullet t \wedge t' \in E_1(M,s)\} \cup & \\ \{\langle t, \langle t,s\rangle\rangle \mid t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s)\} \cup & \\ \{\langle\langle t,s,i\rangle, t'\rangle \mid & t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s) \wedge \\ & i \in \{2,3\} \wedge t' \in E_i(M,t,s)\} \cup \\ \{\langle t, s\rangle \mid t \in T \wedge \neg M[t\rangle \wedge s \in {}^\bullet t\} \cup & \\ \{\langle\langle t,s\rangle, \langle t,s,i\rangle\rangle \mid t \in T \wedge M[t\rangle \wedge s \in {}^\bullet t \wedge W(s,t) > W(t,s) \wedge i \in \{2,3\}\}. & \end{array}$$

A set $V_s \subseteq V_{AND} \cup V_{OR}$ is legal *iff*

$$\begin{array}{l} (\forall x \in V_s \cap V_{AND}\ \forall y \in V_{AND} \cup V_{OR}\ \langle x,y\rangle \in A \Rightarrow y \in V_s), \\ (\forall x \in V_s \cap V_{OR}\ \exists y \in V_s\ \langle x,y\rangle \in A),\ \textit{and} \\ \exists \tau \in V_s \cap T\ M[\tau\rangle \wedge (\forall s \in {}^\bullet\tau\ E_4(s) \subseteq V_s). \end{array}$$

Lemma 3.26 *The set of transitions of any legal set is stubborn. For each stubborn set, there exists a legal set such that the set of transitions of the legal set is the stubborn set. The set of vertices of the and/or-graph is legal iff the marking is nonterminal.*

Proof. The above results follow trivially from Definitions 3.20 and 3.25. □

The result in Lemma 3.26 is due to Valmari [71, 73].

The deletion algorithm can be described as follows. Let the net be finite and the marking nonterminal. Let's use the terms of Definition 3.25 though it is not necessary to construct any explicit and/or-graph. Only such vertex that is both accessible from some enabled transition and backwards accessible from some enabled transition by the reflexive-transitive closure of A has to be explicitly presented if the and/or-graph is constructed. The set of vertices of the and/or-graph is a legal set, maximal with respect to set inclusion, that contains all enabled transitions. Let then V_s be a legal set, maximal with respect to set inclusion, such that the set of transitions in V_s is a stubborn set T_s. Let $t \in T_s$ be enabled. An attempt to remove t from V_s is performed by starting from t, moving backwards in the and/or-graph, and attempting to remove those vertices that must be removed to get a legal subset, maximal with respect to set inclusion, of $V_s \setminus \{t\}$. When such vertex is encountered that does not have to be removed, the backward search is not continued from the vertex. The special condition related to E_4 (the last condition in the definition of legal sets) is not checked until the cumulative removal attempt is over. If the set of vertices that were not attempted to remove from V_s is legal, the cumulative removal attempt is realized. The set of transitions in the new legal set is then a stubborn subset, maximal with respect to set inclusion, of $T_s \setminus \{t\}$. On the other hand, if the attempt to remove t from V_s fails, it is not possible to remove t from any subset of V_s either. The deletion algorithm proceeds by repeatedly attempting to remove a new enabled transition from the current legal set and realizing each successful attempt. The set of vertices of the and/or-graph is the initial legal set. The algorithm stops when it is no longer possible to remove any enabled transition from the current legal set. The set of transitions of the final legal set is minimal in the sense that no proper subset of its enabled transitions can be the set of enabled transitions of any stubborn set. This result is based on what is stated above about the set of transitions of the current legal set before and after removing an enabled transition.

The time taken by an execution of this algorithm is at most proportional to $\mu\nu|T|^2$, where μ is the maximum number of input places of a transition, and ν is the maximum number of adjacent transitions of a place. Our deletion algorithm has no factor that would make it more than a constant times slower than the deletion algorithm in [71].

3.6 Stubborn Set Method and High-Level Nets

Place/transition nets of actual systems tend to be very large. On the other hand, using *high-level nets* [8, 43] one can make compact models in a natural way. Fortunately, a high-level net can often be *unfolded* into a behaviourally equivalent finite place/transition net, and, using the inverse mapping of the unfolding mapping, the place/transition net can be folded back into the high-level net [24, 40]. If such unfolding exists, one can apply the stubborn set method to the result of the unfolding and then fold the reduced reachability graph. However, even a high-level net, the set of reachable markings of which is finite, may be difficult to unfold since the unfolding procedure usually needs explicit bounds on the possible transition instances. If suitable bounds are not known, these have to be estimated.

Unfolding can be implicit in the sense that no place/transition net is constructed but the transition instances of the high-level net are used just like transitions of a place/transition net. Implicit unfolding supports determining the enabled transition instances by unifying the arc expressions with the current marking.

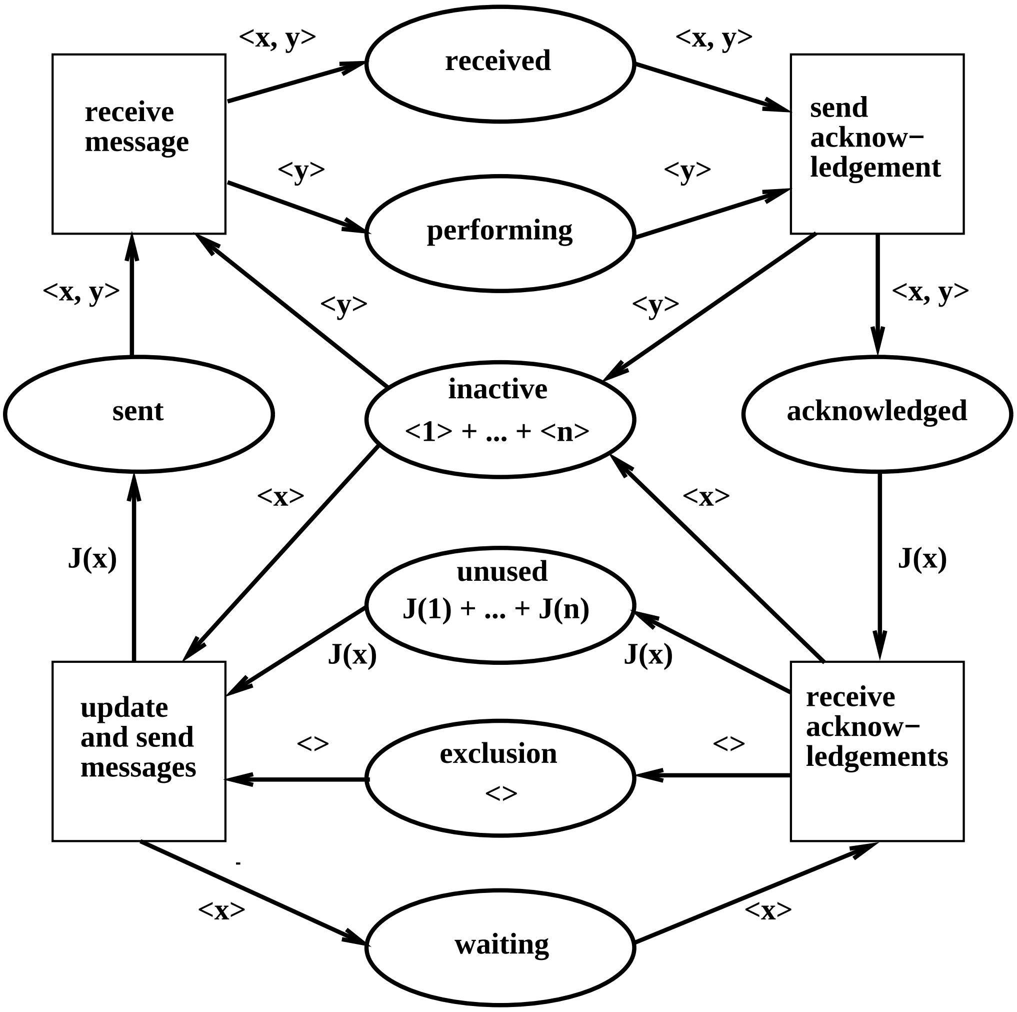
Valmari has presented the stubborn set method in coloured Petri nets [40, 41, 78]. He mentions the possibility to ignore the colour information but states [78]: "Some preliminary experiments have demonstrated that ignoring the colour information usually leads to grossly unnecessarily large stubborn sets." He then turns to implicit unfolding.

The need for unfolding can be reduced to the need to use disabled transition instances in the computation of a stubborn set in a high-level net. If there were a practical algorithm for computing the set of enabled transition instances of a stubborn set without using disabled transitions, we would only have to determine the enabled transition instances and apply the algorithm. The set of disabled transition instances could then be arbitrarily large, even infinite. We shall not, however, pursue this matter further, but leave it for future research to resolve.

3.7 On Choosing a Scapegoat in the Incremental Algorithm

As mentioned at the beginning of Section 3.5, the stubborn sets computed by the incremental algorithm may contain unnecessarily many enabled transitions. As the next example shows, the problem is serious.

Figure 12 presents a data base system of $n \geq 2$ data base managers. The predicate/transition net [24] in the figure is equivalent to the coloured Petri net in [40]. The contained resource allocation and a great amount of concurrency makes the system inherently very suitable for the stubborn set method. Let's assume the most obvious unfolding [24] into a place/transition net with infinite capacities. The image of a transition instance of the predicate/transition net is a transition of the place/transition net, the image of a place-tuple pair of the predicate/transition net is a place of the place/transition net, etc. The resulting place/transition net has no self-loop. The full reachability graph of the place/transition net has $n \cdot 3^{n-1} + 1$ vertices and $2n(1 + (n-1) \cdot 3^{n-2})$ edges [73, 75]. The stubborn set method is capable of producing a reduced reachability graph having $2n^2 - n + 1$ vertices and $2n^2$ edges [73, 75]. In the reduced reachability graph in question (unique up to isomorphism),



(J(x’) = J(x’, 1) + ... + J(x’, n−1))
(J(x’, i) = <x’, ((x’−1+i) mod n)+1>)

Figure 12: A data base system.

the vertex corresponding to the initial marking has n immediate successors. Every other vertex has one and only one immediate successor. From now on, this reduced reachability graph will be called the Λ-graph.

We shall now investigate the behaviour of the incremental algorithm in the above place/transition net. Clearly, the net has no self-loop, so Definition 3.20 implies that for each marking M, transition t, and place s, $E_2(M,t,s) = E_4(s)$. The function E_2 is thus certainly preferable to the function E_3 in the incremental algorithm as far as this net is concerned. So we choose the corresponding choice function b (that occurs in Definition 3.23) to be the function that has the value 2 everywhere.

We shall present four scapegoat generators, called α, β, γ, and δ. The scapegoat generator α is a pathological scapegoat generator not leading to any reduction. The scapegoat generators β, γ, and δ look much like α but γ and δ lead to the Λ-graph, and β leads close to the size of the Λ-graph. In addition, pseudo-random scapegoat generators are considered. They tend to be almost as bad as α.

The scapegoat generator α is defined as follows: $\alpha(M',t')$ is defined iff the transition t' is disabled at the marking M'. Let a transition t be disabled at a marking M. Let $\ell(M,t,p_1,\ldots,p_j)$ denote the first element in a place list $p_1,\ldots,p_j$ that is a disabling place of t at M. Then

$$\alpha(M,t) = \begin{cases} \ell(M,t,\ \langle x'\rangle_{\text{inactive}}, \langle\rangle_{\text{exclusion}}, (J(x',1))_{\text{unused}}, \ldots, (J(x',n-1))_{\text{unused}}) \\ \qquad\quad \text{if } t = \langle x'\rangle_{\text{update and send messages}}, \\ \ell(M,t,\ (J(x',1))_{\text{acknowledged}}, \ldots, (J(x',n-1))_{\text{acknowledged}}, \langle x'\rangle_{\text{waiting}}) \\ \qquad\quad \text{if } t = \langle x'\rangle_{\text{receive acknowledgements}}, \\ \ell(M,t,\langle y'\rangle_{\text{inactive}}, \langle x',y'\rangle_{\text{sent}}) \qquad \text{if } t = \langle x',y'\rangle_{\text{receive message}}, \\ \ell(M,t,\langle y'\rangle_{\text{performing}}, \langle x',y'\rangle_{\text{received}}) \quad \text{if } t = \langle x',y'\rangle_{\text{send acknowledgement}}. \end{cases}$$

Note that $\langle x',y'\rangle_{\text{receive message}}$ corresponds to the instance of the high-level transition "receive message" with $x = x'$ and $y = y'$ whereas $\langle x',y'\rangle_{\text{sent}}$ corresponds to the pair of the high-level place "sent" and tuple $\langle x',y'\rangle$. The other indexed tuples are analogous.

It is straightforward to show that for each marking M that is reachable from the initial marking and for each transition t that is enabled at M, $E^*_{14}(\alpha,b,M,t)$ contains all transitions that are enabled at M. This means that the incremental algorithm (optimized or not) has no reductive effect. The rotation from s to the next possible manager in the definition of $\alpha(M, \langle x'\rangle_{\text{receive acknowledgements}})$ is the actual pathological property of α. This kind of stepping from a manager to another manager occurs very often when a pseudo-random scapegoat generator is used. As a result, pseudo-random generators tend to be almost as bad as α.

The pathological property of α can be eliminated by using a fixed manager when possible. Let $H(x',i)$ be $\langle x',i\rangle$ if $x' > i$, and $\langle x',i+1\rangle$ if $x' \leq i$. The scapegoat generator β is defined as α with the following exception:

$$\beta(M,t) = \ell(M,t,(H(x',1))_{\text{acknowledged}},\ldots,(H(x',n-1))_{\text{acknowledged}},\langle x'\rangle_{\text{waiting}})$$

if $t = \langle x'\rangle_{\text{receive acknowledgements}}$ and t is disabled at M.

When the incremental algorithm (optimized or not) and β are used, the reduced reachability graph has $4(n-2)$ vertices and $8(n-2)$ edges more than the Λ-graph. On the branches where manager 1 or 2 is waiting for acknowledgements, each vertex has exactly one immediate successor. On each of the other $n-2$ branches, the number of vertices having exactly two immediate successors is four, and every other vertex on the branch has exactly one immediate successor. It is rather straightforward to prove these results.

In the scapegoat generator γ, the places representing the phases of the managers have the highest priority. The scapegoat generator γ is defined as α with the following exception:

$$\gamma(M,t) = \ell(M,t,\langle x'\rangle_{\text{waiting}},(J(x',1))_{\text{acknowledged}},\ldots,(J(x',n-1))_{\text{acknowledged}})$$

if $t = \langle x'\rangle_{\text{receive acknowledgements}}$ and t is disabled at M.

The scapegoat generator δ has been got by considering that if a transition has a unique characteristic input place, then that place should have the highest priority. The scape goat generator δ is not pure in that sense but shows the sufficient interchange to transform α into an optimal scapegoat generator. The scapegoat generator δ is defined as α with the following exception:

$$\delta(M,t) = \ell(M,t,\langle x',y'\rangle_{\text{received}},\langle y'\rangle_{\text{performing}})$$

if $t = \langle x',y'\rangle_{\text{send acknowledgement}}$ and t is disabled at M.

It is straightforward to show that for each non-initial marking M that is reachable from the initial marking and for each transition t that is enabled at M, the set of enabled transitions in $E^*_{14}(\gamma,b,M,t)$ is $\{t\}$, and the set of enabled transitions in $E^*_{14}(\delta,b,M,t)$ is $\{t\}$. As a consequence, the incremental algorithm (optimized or not) produces the Λ-graph.

The above example suggests some heuristics for choosing a scapegoat. The scapegoat generator β suggests absolute ordering with respect to identity numbers, the scapegoat generator γ suggests giving a process control place the highest priority, and the scapegoat generator δ suggests giving a unique characteristic input place the highest priority. All these strategies are fixed order strategies in the sense that always the first possible alternative in a fixed list is chosen.

Some strategies work without knowledge of the modelled system. One of such strategies minimizes the number of enabled immediate successors of a vertex that are not in any maximal strongly connected component already found in the dependency graph. On the second priority level, it minimizes the number of all immediate successors of the vertex that are not in any maximal strongly connected component already found. On the third priority level, it minimizes the number of those immediate successors of the vertex that have not been visited yet.

A pseudo-random scapegoat generator is probably far from optimal if the incremental algorithm is as instable as in the above example. On the other hand, it is often useful to compare a given strategy to a pseudo-random strategy to see how good the strategy is. A pseudo-random strategy is a good measuring stick since it employs no knowledge of the system. If a strategy gives better (worse) results than a pseudo-random strategy, there must be something good (bad) in the strategy.

The deletion algorithm can also be used to estimate how good the incremental algorithm could be even though the deletion algorithm may sometimes compute a stubborn set containing more enabled transitions than a stubborn set computed by the incremental algorithm. The choice of an enabled transition to be deleted is a nondeterministic factor. In the case of the net in Figure 12, the nondeterminism associated with the deletion algorithm does not affect the number of enabled transitions in the computed stubborn set.

4 Sleep Set Method

In this chapter we present Godefroid's *sleep set method* [25, 27, 28, 30, 26, 36, 86, 87]. The plain sleep set method preserves at least one sequence from each conditional trace leading from the initial state to a terminal state. To prevent a transition from firing, it is put into a so called sleep set.

Wolper and Godefroid [86], Godefroid and Pirottin [28], and Wolper, Godefroid, and Pirottin [87] have combined the sleep set method with the stubborn set method. The combination is justified by the fact that the stubborn set method alone is sometimes bound to fire independent transitions at a state. The combination presented in [28, 86, 87] is such that at each encountered nonterminal state a nonempty persistent set is computed. Let us recall from Lemma 3.12 that a set is a nonempty persistent set iff the set is a strongly dynamically stubborn set consisting of enabled transitions only. In [86, 87], persistence is defined on the basis of global independence but since global independence implies independence at each reachable state, the persistent sets in [86, 87] are persistent in the sense defined by Godefroid and Pirottin [28]. As mentioned immediately above Definition 3.9, our definition of persistence in Definition 3.9 corresponds to Godefroid's and Pirottin's definition [28]. The plain sleep set method can be thought as a special case of the combined method: a simple heuristic for computing a persistent set is used. We shall not consider the plain sleep set method further.

Section 4.1 presents an algorithm for detecting reachable terminal states in place/transition nets. The practicalities related to the algorithm are considered in Section 4.2.

4.1 A Terminal Marking Detection Algorithm

In this section, we concentrate on a generalized version of Wolper's and Godefroid's terminal state detection algorithm [86]. The generalized version is in Figure 13. The intuitive idea of the algorithm is to eliminate such redundant *interleavings of transitions* that are not eliminated by the transition selection function f. We show that the algorithm is guaranteed to find all reachable terminal markings of any finite place/transition net with a finite set of reachable markings. Any dynamically stubborn function is valid for f, but the algorithm is not limited to dynamically stubborn sets. It suffices that f represents all sets of length-secure alternative sequences to terminal markings. The set T_0 can be any subset of transitions that are disabled at the initial marking. The φ in Figure 13 can be any truth-valued function on $\mathcal{M} \times T \times T \times 2^T$ that satisfies: if $\varphi(M, t, t', T_s)$, then either t and t' commute at M and $t' \in T_s$, or tt' is disabled at M. For example, $\varphi(M, t, t', T_s)$ could be

- "either t and t' commute at M and $t' \in T_s$, or tt' is disabled at M",

```
make Stack empty; make $H$ empty;
push $\langle M_0, T_0 \rangle$ onto Stack;
while Stack is not empty do {
    pop $\langle M, \text{Sleep} \rangle$ from Stack;
    if $M$ is not in $H$ then {
        Fire $= \{t \in f(M) \setminus \text{Sleep} \mid M[t\rangle\}$;
        if Fire and Sleep are both empty then print "Terminal state!";
        enter $\langle M, \text{a copy of Sleep} \rangle$ in $H$;
    }
    else {
        let hSleep be the set associated with $M$ in $H$;
        Fire $= \{t \in \text{hSleep} \setminus \text{Sleep} \mid M[t\rangle\}$;
        Sleep $=$ hSleep $\cap$ Sleep;
        substitute a copy of Sleep for the set associated with $M$ in $H$;
    }
    for each $t$ in Fire do {
        let $M[t\rangle M'$;
        tSleep $= \{t' \in T \mid \varphi(M, t, t', \text{Sleep})\}$;
        push $\langle M', \text{a copy of tSleep} \rangle$ onto Stack;
        Sleep $= \{t\} \cup$ Sleep;
    }
}
```

Figure 13: A terminal marking detection algorithm.

- "t and t' are independent at M and $t' \in T_s$",
- "t and t' commute at M and $t' \in T_s$", or simply
- "false".

Note that if $M[t\rangle M'$, then tt' is disabled at M iff t' is disabled at M'. So the first alternative in the above list has the effect that if t is fired from M to M' in the algorithm in Figure 13, then the sleep set pushed onto the stack with M' contains all those transitions that are disabled at M'. We shall consider the practicalities related to φ and T_0 in Section 4.2.

The algorithm in Figure 13 is similar to Wolper's and Godefroid's algorithm [86]. The only essential differences are that Wolper and Godefroid assume that the set corresponding to $f(M)$ is persistent, the set corresponding to T_0 is empty, and the condition corresponding to $\varphi(M, t, t', T_s)$ is "t and t' are globally independent and $t' \in T_s$".

Theorem 4.1 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a finite place/transition net such that the set of markings reachable from M_0 is finite. Let f be a function from $\mathcal{M}$ to 2^T such that f represents all sets of length-secure alternative sequences to terminal markings. Let T_0 be a subset of transitions that are disabled at M_0. Let φ be a truth-valued function on $\mathcal{M} \times T \times T \times 2^T$ such that for each marking M, for all transitions t and t', and for each $T_s \subseteq T$, if $\varphi(M, t, t', T_s)$, then either t and t' commute at M and $t' \in T_s$, or tt' is disabled at M. Then the algorithm in Figure 13 finds all terminal markings that are reachable from M_0.*

Proof. Slight modifications of the proof of Theorem 6 in [86] suffice. There are only two essential differences. The first essential difference is that if we are visiting M, and a finite transition sequence σ leads from M to a terminal marking, the set of length-secure alternative sequences of σ at M, $\vartheta(\sigma, M)$, is used in the reasoning instead of the trace of σ. The second essential difference is that instead of independence, the weaker property satisfied by φ is used. The modified proof concentrates on showing that if no length-secure alternative sequence of a finite transition sequence leading from a marking of a stack element to a terminal marking has its first transition in the sleep set of the stack element, the terminal marking is visited. This is sufficient since at the time of entering the "while-loop", the stack contains only the initial marking with a sleep set containing disabled transitions only. In the proof, an arbitrary terminal marking that is reachable from the initial marking is considered, and induction is used on the length of a finite transition sequence leading from a marking on the stack to the terminal marking. Since the length-secure alternative sequences of a finite transition sequence σ are of length less than or equal to the length of σ, the induction hypothesis is of the form "for each $k \leq n$" instead of "for n". The above property of φ is needed in showing that if $M[t\rangle M'$, $M'[t'\rangle M''$, t is fired at M, and t' is in the sleep set pushed onto the stack with M' when t is fired at M, then $t' \in$ Sleep at the time of the push and $M[t't\rangle M''$. □

The result in Theorem 4.1 is new though inspired by Wolper and Godefroid [86], Godefroid and Pirottin [28], and Wolper, Godefroid, and Pirottin [87].

Let us recall from Theorem 3.16 that dynamically stubborn functions represent all sets of enabled permutations to terminal markings. So they represent all sets of length-secure alternative sequences to terminal markings, too. From Theorem 4.1 it thus follows that the algorithm in Figure 13 is compatible with all dynamically stubborn sets.

The stack in the algorithm in Figure 13 could be replaced by a set, a queue, or a priority queue since a set, a queue, as well as a priority queue would suffice in the proof of Theorem 4.1. Wolper and Godefroid do not utilize the characteristics of stacks in their corresponding proof either [86]. Unfortunately, there seems to be no easy way to find the reachable terminal markings in the order of shortest distance from the initial marking since new transitions can be fired when a marking "near the initial marking" is revisited. By the shortest distance we mean the shortest

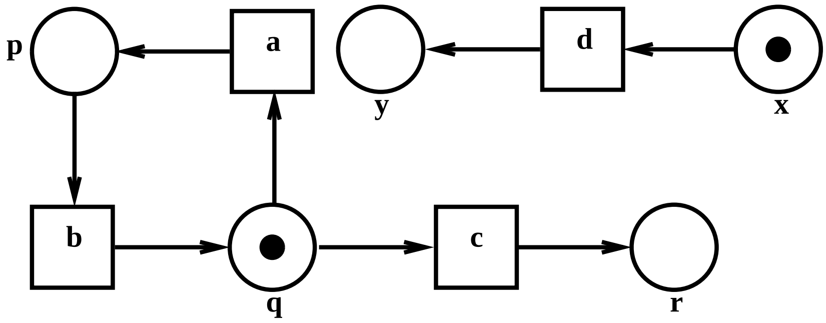


Figure 14: A deceptive f makes the algorithm in Figure 13 fail.

distance with respect to the paths of the full reachability graph traversed during the execution of the algorithm.

The proof of Theorem 4.1 takes advantage of the possibility to fire new transitions when a marking is revisited. On the other hand, we do not currently know whether it is possible to improve the algorithm in Figure 13 in such a way that no marking would be visited more than once but all reachable terminal markings would still be found. We do not either know whether it is possible to weaken the conditions for f and φ in the algorithm in Figure 13 in such a way that all reachable terminal markings would still be found. However, the next example suggests that our condition for f may at least be a good approximation of the weakest condition for f if such condition exists.

Let's consider an example which shows that the statement obtained from Theorem 4.1 by removing the word "length-secure" is not valid. Let $\varphi(M, t, t', T_s)$ iff t and t' commute at M and $t' \in T_s$. Let $T_0 = \emptyset$. Let $M_0[a\rangle M_1$, $M_1[d\rangle M_2$, and $M_2[bc\rangle M_3$ in the net in Figure 14. Let f be defined by $f(M_0) = \{a\}$ and $f(M) = T$ when $M \neq M_0$. For each $M \in \mathcal{M}$, if M_0 is reachable from M, then $M = M_0$ or $M = M_1$. The marking M_3 is the only terminal marking that is reachable from M_0. The function f thus represents all sets of alternative sequences to terminal markings. However, f does not represent all sets of length-secure alternative sequences to terminal markings since $M_0[cd\rangle M_3$ but for each $\sigma \in T^*$ of length less than or equal to 2, $\neg M_0[\sigma\rangle_f M_3$.

During the first visit to M_0, the algorithm in Figure 13 inserts $\langle M_0, \emptyset\rangle$ into H and pushes $\langle M_1, \emptyset\rangle$ onto the stack. The algorithm then visits M_1. The transitions b and d are the enabled transitions in $f(M_1) = T$ at M_1. Let b be fired before d at M_1. The algorithm pushes $\langle M_0, \emptyset\rangle$ and $\langle M_2, \{b\}\rangle$ onto the stack since b and d commute at M_1. The algorithm then visits M_2 but does not fire the sleeping b which is the only enabled transition at M_2. No transition is fired during the second visit to M_0 since the sleep set associated with M_0 in H is empty. The execution of the algorithm is then over. No terminal marking was found though M_3 is a terminal marking that is reachable from M_0.

It is possible to extend the algorithm in Figure 13 and Theorem 4.1 to high-level nets [8, 43]. The most straightforward way to do this is to turn each transition into a transition instance. The computation of $f(M)$ is the only part of the algorithm that may require knowledge about the bounds on the set of possible transition instances.

Though the models of concurrency in [28, 86] are more sophisticated than place/transition nets, we claim that a result equivalent to Theorem 4.1 should hold for those models. We justify the claim as follows.

- The concept of an alternative sequence of a finite transition sequence at a state is general and suits the models of concurrency in [28, 86] perfectly.
- In those models, transitions t and t' commute at a state M iff tt' and $t't$ are enabled at M and $t't$ leads from M to the same state as tt'. (In place/transition nets, if tt' and $t't$ are enabled at a marking M, then $t't$ leads from M to the same marking as tt'.)
- The algorithm in Figure 13 suits those models as such.
- The proof of Theorem 4.1 can thus essentially be repeated for those models.

4.2 Practicalities

The practicalities related to the algorithm presented in Section 4.1 are considered in this section. We start by considering the practicalities related to the φ and the T_0 in the algorithm in Figure 13.

Lemma 4.2 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a finite place/transition net. Let φ be a truth-valued function on $\mathcal{M} \times T \times T \times 2^T$ such that for each marking M, for all transitions t and t', and for each $T_s \subseteq T$, if $\varphi(M, t, t', T_s)$, then t and t' are independent at M and $t' \in T_s$. Let $T_0 = \emptyset$. Then in the algorithm in Figure 13, each sleep set associated with a marking contains only transitions that are enabled at the marking.*

Proof. If a transition t' is in the sleep set pushed onto the stack with a marking M' when a transition t is fired from a marking M to M', then $t' \in$Sleep at the time of the push, and each transition in Fire is enabled at M. The sleep set associated with the initial marking at the beginning of the execution of the algorithm is empty. If transitions t and t'' are enabled and independent at a marking M, and $M[t\rangle M''$, then t'' is enabled at M''. The result thus follows by a trivial induction. □

The result in Lemma 4.2 is due to Wolper and Godefroid [86] despite the differences between the algorithm in Figure 13 and their terminal state detection algorithm.

```
make Stack empty; make $H$ empty;
push $\langle M_0, \emptyset \rangle$ onto Stack;
while Stack is not empty do {
    pop $\langle M, \text{Sleep} \rangle$ from Stack;
    if $M$ is not in $H$ then {
        Fire$=\{t \in f(M) \setminus \text{Sleep} \mid M[t\rangle\}$;
        if Fire and Sleep are both empty then print "Terminal state!";
        enter $\langle M, \text{a copy of Sleep} \rangle$ in $H$;
    }
    else {
        let hSleep be the set associated with $M$ in $H$;
        Fire$=$hSleep$\setminus$Sleep;
        Sleep$=$hSleep$\cap$Sleep;
        substitute a copy of Sleep for the set associated with $M$ in $H$;
    }
    for each $t$ in Fire do {
        let $M[t\rangle M'$;
        tSleep$=\{t' \in \text{Sleep} \mid t$ and $t'$ commute at $M\}$;
        push $\langle M', \text{a copy of tSleep} \rangle$ onto Stack;
        Sleep$=\{t\} \cup$Sleep;
    }
}
```

Figure 15: A practical implementation of the algorithm in Figure 13 with respect to φ and T_0.

In Figure 15, an implementation of the algorithm in Figure 13 with respect to φ and T_0 is presented. The algorithm in Figure 15 can be obtained from the algorithm in Figure 13 by making T_0 empty, removing the checking of enabledness from the "else-block", and defining: $\varphi(M, t, t', T_s)$ iff t and t' commute at M and $t' \in T_s$. We know that transitions commute at a marking iff they are enabled and independent at the marking. Checking commutation should be easier than checking independence. Lemma 4.2 implies that the algorithm in Figure 13 is equivalent to the algorithm in Figure 15 when T_0 is empty and φ is defined: $\varphi(M, t, t', T_s)$ iff t and t' commute at M and $t' \in T_s$. Lemma 15 thus also implies that in the algorithm in Figure 15, each sleep set associated with a marking contains only transitions that are enabled at the marking.

Lemma 4.3 *Let $\langle S, T, F, K, W, M_0 \rangle$ be a finite place/transition net such that the set of markings reachable from M_0 is finite. Let T_0 be a subset of transitions that* *are disabled at M_0. Let φ be a truth-valued function on $\mathcal{M} \times T \times T \times 2^T$ such that for each marking M, for all transitions t and t', and for each $T_s \subseteq T$, if $\varphi(M, t, t', T_s)$, then either t and t' commute at M and $t' \in T_s$, or tt' is disabled at M. Let's further require that for each marking M, for all transitions t and t', and for each $T_s \subseteq T$, if t and t' commute at M and $t' \in T_s$, then $\varphi(M, t, t', T_s)$. Let's assume that the sets, the set operations (insertion, union, intersection, and difference), the stack, the stack operations, the "for-loop", and the computation of $f(M)$ in the algorithms in Figure 13 and 15 are implemented exactly in the same way. Then the algorithms visit exactly the same markings and fire exactly the same transitions in exactly the same order.*

Proof. Let's assume that a transition t is being fired from a marking M to a marking M' in the algorithm in Figure 13. If a transition t' is enabled at M' and is in the sleep set pushed onto the stack with M' when t is fired at M, then $\varphi(M, t, t', \text{Sleep})$ holds at the time of the push but tt' is enabled at M, so t and t' commute at M, $t' \in \text{Sleep}$ at the time of the push, and t' is enabled at M. The result now follows by a trivial induction. □

The result in Lemma 4.3 is new though inspired by Wolper and Godefroid [86].

Lemma 4.3 states that there is no more refined implementation of the algorithm in Figure 13 with respect to φ and T_0 than the algorithm in Figure 15 if φ and T_0 are required to satisfy the assumptions in Theorem 4.1. Lemmas 4.2 and 4.3 suggest the heuristic that each sleep set associated with a marking should only contain transitions that are enabled at the marking.

Let's consider the complexity of the algorithm in Figure 15. The time taken by a check of whether two transitions commute at a marking is at most proportional to ν, where ν is the maximum number of adjacent places of a transition. The cumulative time per marking spent in the "for-loop" is at most proportional to $\nu\rho^2$, where ρ is the maximum number of enabled transitions of a marking, and all visits to the marking are counted. This is based on the fact that each sleep set associated with a marking contains only transitions that are enabled at the marking. The time per visit to a marking spent in the operations related to H is the time of the search for the marking plus a time that is at most proportional to ρ. The searches in H are something that cannot be avoided easily whether or not we use sleep sets at all. It depends much on the net how many times a marking is visited and how many simultaneous occurrences of a marking there are in the stack. One stack element requires space for the marking and at most ρ transitions. It is not necessary to store copies of markings and transitions since pointers suffice. More clever ways to cut down on space consumption in sleep set algorithms have been presented by Godefroid, Holzmann, and Pirottin [26].

The combination of the sleep set method and the stubborn set method can really be better than the plain stubborn set method as far as the number of inspected markings

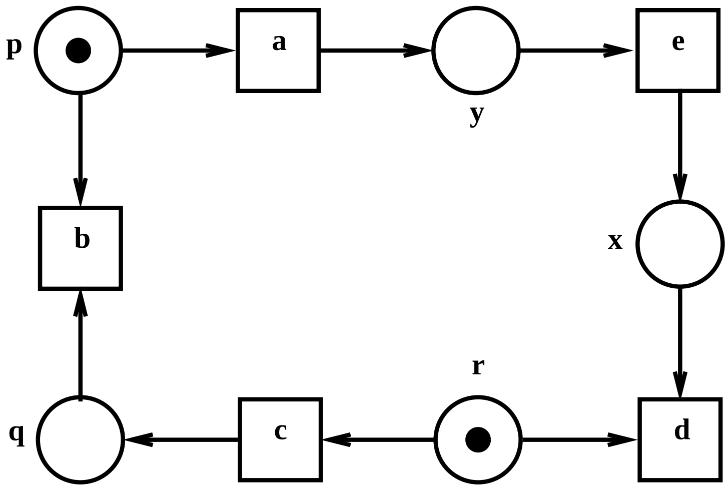


Figure 16: A net showing some of the power of the algorithm in Figure 15.

is concerned. More precisely, there can be a dynamically stubborn function f such that $f(M)$ can be computed by using a feasible algorithm such as the incremental algorithm, and for each dynamically stubborn function g, the number of vertices in the g-reachability graph is greater than the number of markings that are inspected by the algorithm in Figure 15 that uses f. The net in Figure 16 is a simple example showing this. The example is essentially the same as can be found in [87].

An exhaustive investigation shows that at each reachable nonterminal marking of this net, there is one and only one dynamically stubborn set that is minimal with respect to set inclusion. By Lemma 3.8 we know that a dynamically stubborn set that is minimal with respect to set inclusion only contains enabled transitions. Another exhaustive investigation shows that at each reachable nonterminal marking of this net, the set of enabled transitions of any stubborn set computed by the incremental algorithm, using our definition or any of the definitions of stubbornness in [70, 75, 78], is a dynamically stubborn set that is minimal with respect to set inclusion. If M is a nonterminal marking, let $f(M)$ be the set of enabled transitions of a stubborn set computed by the incremental algorithm. Then $f(M)$ is the only dynamically stubborn set at M that is minimal with respect to set inclusion. Thus, for each dynamically stubborn function g, the f-reachability graph is a subgraph of the g-reachability graph.

We have $f(M_0) = \{a, c\}$. Let a be fired before c at M_0 in the algorithm in Figure 15. Let $M_0[c\rangle M'$. Since a and c commute at M_0 and a is fired before c, $\langle M', \{a\}\rangle$ is pushed onto the stack. Let's consider the visit to M' where $\langle M', \{a\}\rangle$ is popped from the stack. We have $f(M') = \{a, b\}$, but the sleeping a is not fired. Let $M'[a\rangle M''$. By executing the algorithm in Figure 15 completely, we see that M'' is never encountered, and no nonterminal marking is visited more than once. The latter observation is important since it guarantees that all transitions that are fired at a marking M are in $f(M)$. The set of inspected markings is thus a proper subset of the markings of the f-reachability graph.

The statistics in [28, 86, 87] concerning some analyzed protocols do not give direct information for comparing the stubborn set method with the combination of the sleep set method and the stubborn set method.

5 Conclusions

We have studied Valmari's stubborn set method [70, 71, 73, 74, 75, 76, 78, 79, 80] and Godefroid's sleep set method [25, 27, 28, 30, 26, 36, 86, 87] and their combination [28, 86, 87] in place/transition nets [64, 65].

We have shown dynamically stubborn sets [63, 78] to be a useful generalization of stubborn sets. Dynamically stubborn sets seem to have all the nice properties of stubborn sets except that the definition of dynamic stubbornness does not seem to imply a practical algorithm for computing dynamically stubborn sets.

We have chosen a weak definition of stubbornness [70] and presented the incremental algorithm [70, 73] and the deletion algorithm [71, 73] for computing such stubborn sets. The deletion algorithm is guaranteed to find a minimal stubborn set in the sense that no proper subset of its enabled transitions can be the set of enabled transitions of any stubborn set. The time taken by an execution of the incremental algorithm is at most proportional to $\mu\nu|T|$, where μ is the maximum number of input places of a transition, ν is the maximum number of adjacent transitions of a place, and T is the set of transitions of the net. Respectively, the time taken by an execution of the deletion algorithm is at most proportional to $\mu\nu|T|^2$. However, the incremental algorithm often pays the price in the size of the computed stubborn set and consequently in the size of the reduced reachability graph. The deletion algorithm can be used to estimate how good the incremental algorithm could be even though the deletion algorithm may sometimes compute a stubborn set containing more enabled transitions than a stubborn set computed by the incremental algorithm.

The choices made about of disabling places (scapegoats) in the incremental algorithm can seriously affect the number of enabled transitions in the resulting stubborn set. We considered an example that suggests three fixed order strategies for choosing a scapegoat: absolute ordering with respect to identity numbers, giving a process control place the highest priority, and giving a unique characteristic input place the highest priority. It is also easy to develop simple strategies that work without knowledge of the modelled system. To see how good a given strategy is, one can compare it to a pseudo-random strategy.

The known practical algorithms for computing stubborn sets in high-level nets [78] use disabled transition instances. Consequently, the high-level net is unfolded into a finite place/transition net at least implicitly during analysis. Unfolding requires that if suitable bounds for the possible transition instances are not known, these have to be estimated.

The stubborn set method alone is sometimes bound to fire independent transitions at a state, so the sleep set method can further be used to eliminate redundant interleavings of transitions. We have generalized Wolper's and Godefroid's terminal state detection algorithm [86] and shown that the generalized version detects all reachable terminal markings of any finite place/transition net with a finite full reachability graph, given that the transition selection function represents all sets of length-secure alternative sequences to terminal markings. As already known, dynamically stubborn functions represent all sets of enabled permutations to terminal markings. They thus also represent all sets of length-secure alternative sequences to terminal markings.

The contributions of this thesis are the following:

- We have generalized Wolper's and Godefroid's terminal state detection algorithm [86]. We have shown that the generalized version detects all reachable terminal markings of any finite place/transition net with a finite full reachability graph if the transition selection function represents all sets of length-secure alternative sequences to terminal markings. We have also given justifications to our claim that an equivalent result should hold in the models of concurrency of Wolper and Godefroid [86], and Godefroid and Pirottin [28].
- We have found heuristics for choosing a scapegoat in the incremental algorithm which computes stubborn sets. Bad choices of scapegoats can seriously affect the number of states explored in a reachability analysis.
- We have proven that conventionally dynamically stubborn functions represent all conditional traces to terminal markings.
- We have presented Godefroid's and Pirottin's definitions of persistence and conditional stubbornness [28] in the context of place/transition nets. For these, we have shown that a set is conditionally stubborn iff it is strongly dynamically stubborn. Consequently, a set is a nonempty persistent set iff the set is a strongly dynamically stubborn set consisting of enabled transitions only.
- We have improved the theory of dynamically stubborn sets by presenting some important results that have been known for long but have missed explicit treatment.
- We have improved the understanding of dynamically and statically stubborn sets by showing some small but interesting results and presenting many counterexamples.

Wolper and Godefroid [86], and Wolper, Godefroid, and Pirottin [87] suggest that the stubborn set method and the sleep set are compatible in a broad area of verification. The compatibility should certainly be studied further since all available means should be utilized in attacking the state space explosion problem, and in our opinion, only the detection of reachable terminal states has obtained more than cursory treatment so far. Linear temporal logics seem to form the most central area of research since they have a great expressive power, and the stubborn set method alone as well as the sleep set method alone can be extended to verify properties expressed as linear temporal logic formulae without a next state -operator [29, 59, 76, 79]. The combination of the stubborn set method and the sleep set method should be studied in all those models of concurrency where each of these two methods alone are applicable. Also, it should be studied whether it is possible to compute suitable stubborn sets in high-level nets without using disabled transition instances, though it may seem that such possibility does not exist. Finally, we have the problem of how efficient the algorithms are in practice and what could be done to improve their efficiency.

References

[1] Ashcroft, E. and Manna, Z.: *Formalization of Properties of Parallel Programs.* Meltzer, B. and Michie, D. (Eds.), Machine Intelligence 6. Edinburgh University Press, Edinburgh 1971, pp. 17–42.

[2] Berthelot, G.: *Transformations and Decompositions of Nets.* In [8], pp. 359–376.

[3] 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.

[4] Best, E. (Ed.): Proceedings of the 4th International Conference on Concurrency Theory, Hildesheim, August 1993. Lecture Notes in Computer Science 715, Springer-Verlag, Berlin 1993, 541 p.

[5] Best, E. and Lengauer, C.: *Semantic Independence.* Arbeitspapiere der GMD 250, Sankt Augustin 1987, 32 p.

[6] Billington, J., Wheeler, G., Keck, B., and Parker, K.: *FORSEE.* Tool descriptions of the 5th International Workshop on Petri Nets and Performance Models, Melbourne, December 1991, 16 p.

[7] von Bochmann, G. and Probst, D.K. (Eds.): Proceedings of the 4th International Workshop on Computer-Aided Verification, Montreal, June 1992. Lecture Notes in Computer Science 663, Springer-Verlag, Berlin 1993, 422 p.

[8] Brauer, W., Reisig, W., and Rozenberg, G. (Eds.): *Petri Nets: Central Models and Their Properties. Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, September 1986.* Lecture Notes in Computer Science 254, Springer-Verlag, Berlin 1987, 480 p.

[9] Brauer, W., Reisig, W., and Rozenberg, G. (Eds.): *Petri Nets: Applications and Relationships to Other Models of Concurrency. Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, September 1986.* Lecture Notes in Computer Science 255, Springer-Verlag, Berlin 1987, 516 p.

[10] Bryant, R.E.: *Graph-Based Algorithms for Boolean Function Manipulation.* IEEE Transactions on Computers C-35 (1986) 8, pp. 677–691.

[11] Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., and Hwang, L.J.: *Symbolic Model Checking:* 10^{20} *States and Beyond.* Information and Computation 98 (1992) 2, pp. 142–170.

[12] Clarke, E.M., Filkorn, T., and Jha, S.: *Exploiting Symmetry in Temporal Logic Model Checking.* In [16], pp. 450–462.

[13] Clarke, E.M. and Kurshan, R.P. (Eds.): Proceedings of the 2nd International Workshop on Computer-Aided Verification, New Brunswick NJ, June 1990. Lecture Notes in Computer Science 531, Springer-Verlag, Berlin 1991, 372 p.

[14] Cleaveland, W.R. (Ed.): Proceedings of the 3rd International Conference on Concurrency Theory, Stony Brook NY, August 1992. Lecture Notes in Computer Science 630, Springer-Verlag, Berlin 1992, 580 p.

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

[16] Courcoubetis, C. (Ed.): Proceedings of the 5th International Conference on Computer-Aided Verification, Elounda, Greece, June/July 1993. Lecture Notes in Computer Science 697, Springer-Verlag, Berlin 1993, 504 p.

[17] Courcoubetis, C., Vardi, M., Wolper, P., and Yannakakis, M.: *Memory Efficient Algorithms for the Verification of Temporal Properties.* Formal Methods in System Design 1 (1992) 2/3, pp. 275–288.

[18] Dams, D., Grumberg, O., and Gerth, R.: *Generation of Reduced Models for Checking Fragments of CTL.* In [16], pp. 479–490.

[19] Dijkstra, E.W.: *Hierarchical Ordering of Sequential Processes.* Hoare, C.A.R. and Perrott, R.H. (Eds.), Operating Systems Techniques, Proceedings of a Seminar held at Queen's University, Belfast 1971. APIC Studies in Data Processing 9, Academic Press, London 1972, pp. 72–93.

[20] Emerson, E.A. and Sistla, A.P.: *Symmetry and Model Checking.* In [16], pp. 463–478.

[21] Fernandez, J.-C., Kerbrat, A., and Mounier, L.: *Symbolic Equivalence Checking.* In [16], pp. 85–96.

[22] Findlow, G.: *Obtaining Deadlock-Preserving Skeletons for Coloured Nets.* In [42], pp. 173–192.

[23] Finkel, A. and Petrucci, L.: *Avoiding State Explosion by Composition of Minimal Covering Graphs.* In [49], pp. 169–180.

[24] Genrich, H.J.: *Predicate/Transition Nets.* In [8], pp. 207–247.

[25] Godefroid, P.: *Using Partial Orders to Improve Automatic Verification Methods.* In [13], pp. 176–185.

[26] Godefroid, P., Holzmann, G.J., and Pirottin, D.: *State Space Caching Revisited.* In [7].

[27] Godefroid, P. and Kabanza, F.: *An Efficient Reactive Planner for Synthesizing Reactive Plans.* Proceedings of AAAI-91, Anaheim CA, July 1991, Vol. 2, pp. 640–645.

[28] Godefroid, P. and Pirottin, D.: *Refining Dependencies Improves Partial-Order Verification Methods.* In [16], pp. 438–449.

[29] Godefroid, P. and Wolper, P.: *A Partial Approach to Model Checking.* Proceedings of the 6th Annual IEEE Symposium on Logic in Computer Science, Amsterdam, July 1991. IEEE Computer Society Press, Los Alamitos CA 1991, pp. 406–415.

[30] Godefroid, P. and Wolper, P.: *Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties.* Formal Methods in System Design 2 (1993) 2, pp. 149–164.

[31] Graf, S. and Steffen, B.: *Compositional Minimization of Finite-State Processes.* In [13], pp. 186–196.

[32] Grönberg, P., Tiusanen, M., and Varpaaniemi, K.: *PROD—A Pr/T-Net Reachability Analysis Tool.* Helsinki University of Technology, Digital Systems Laboratory Report B 11, Espoo 1993, 44 p.

[33] Haddad, S.: *A Reduction Theory for Coloured Nets.* In [43], pp. 399–425.

[34] Holzmann, G.J.: *Algorithms for Automated Protocol Verification.* AT&T Technical Journal 69 (1990) 1, pp. 32–44.

[35] Holzmann, G.J.: *Design and Validation of Computer Protocols.* Prentice Hall, Englewood Cliffs NJ, 1991, 500 p.

[36] Holzmann, G.J., Godefroid, P., and Pirottin, D.: *Coverage Preserving Reduction Strategies for Reachability Analysis.* Proceedings of the 12th International IFIP WG 6.1 Symposium on Protocol Specification, Testing, and Verification, Lake Buena Vista FL, June 1992. IFIP Transactions C-8, North-Holland, Amsterdam 1992, pp. 349–363.

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

[38] Janicki, R. and Koutny, M.: *Using Optimal Simulations to Reduce Reachability Graphs.* In [13], pp. 166–175.

[39] Janicki, R. and Koutny, M.: *Optimal Simulations, Nets and Reachability Graphs.* Rozenberg, G. (Ed.), Advances in Petri Nets 1991. Lecture Notes in Computer Science 524, Springer-Verlag, Berlin 1991, pp. 205–226.

[40] Jensen, K.: *Coloured Petri Nets and the Invariant Method.* Theoretical Computer Science 14 (1981), pp. 317–336.

[41] Jensen, K.: *Coloured Petri Nets.* In [8], pp. 248–299.

[42] Jensen, K. (Ed.): Proceedings of the 13th International Conference on Application and Theory of Petri Nets, Sheffield, June 1992. Lecture Notes in Computer Science 616, Springer-Verlag, Berlin 1992, 398 p.

[43] Jensen, K. and Rozenberg, G. (Eds.): *High-level Petri Nets. Theory and Application.* Springer-Verlag, Berlin 1991, 724 p.

[44] Jonsson, B., Parrow, J., and Pehrson, B. (Eds.): Proceedings of the 11th International IFIP WG 6.1 Symposium on Protocol Specification, Testing, and Verification, Stockholm, June 1991. North-Holland, Amsterdam 1991, 365 p.

[45] Katz, S. and Peled, D.: *Verification of Distributed Programs Using Representative Interleaving Sequences.* Distributed Computing 6 (1992) 2, pp. 107–120.

[46] Katz, S. and Peled, D.: *Defining Conditional Independence Using Collapses.* Theoretical Computer Science 101 (1992) 2, pp. 337–359.

[47] Kaivola, R. and Valmari, A.: *The Weakest Compositional Semantic Equivalence Preserving Nexttime-less Linear Temporal Logic.* In [14], pp. 207–221.

[48] Kemppainen, J., Levanto, M., Valmari, A., and Clegg, M.: *"ARA" Puts Advanced Reachability Analysis Methods Together.* Proceedings of the 5th Nordic Workshop on Programming Environment Research, Tampere, January 1992. Tampere University of Technology, Software Systems Laboratory Report 14, Tampere 1992, 14 p.

[49] Larsen, K.G. and Skou, A. (Eds.): Proceedings of the 3rd International Workshop on Computer-Aided Verification, Aalborg, July 1991. Lecture Notes in Computer Science 575, Springer-Verlag, Berlin 1992, 487 p.

[50] Lin, B. and Newton, A.R.: *Efficient Symbolic Manipulation of Equivalence Relations and Classes.* Proceedings of the International Workshop on Formal Methods in VLSI Design, Miami FL, January 1991, 19 p.

[51] Lin, B. and Newton, A.R.: *Implicit Manipulation of Equivalence Classes Using Binary Decision Diagrams.* Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors, Cambridge MA, October 1991. IEEE Computer Society Press, Los Alamitos CA 1991, pp. 18–25.

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

[53] Mazurkiewicz, A.: *Trace Theory.* In [9], pp. 279–324.

[54] McMillan, K.L.: *Using Unfoldings to Avoid the State Explosion Problem in the Verification of Asynchronus Circuits.* In [7].

[55] Murata, T., Shenker, B., and Shatz, S.M.: *Detection of Ada Static Deadlocks Using Petri Net Invariants.* IEEE Transactions on Software Engineering SE-15 (1989) 3, pp. 314–326.

[56] Nilsson, N.J.: *Principles of Artificial Intelligence.* Springer-Verlag, Berlin 1982, 476 p.

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

[58] Peled, D.: *Sometimes "Some" is as Good as "All".* In [14], pp. 192–206.

[59] Peled, D.: *All from One, One for All: on Model Checking Using Representatives.* In [16], pp. 409–423.

[60] Pnueli, A.: *Applications of Temporal Logic to the Specification and Verification of Reactive Systems: A Survey of Current Trends.* de Bakker, J.W., de Roever, W.-P., and Rozenberg, G. (Eds.), Current Trends in Concurrency, Overviews and Tutorials. Lecture Notes In Computer Science 224, Springer-Verlag, Berlin 1986, pp. 510–584.

[61] Probst, D.K. and Li, H.F.: *Using Partial-Order Semantics to Avoid the State Explosion Problem in Asynchronous Systems.* In [13], pp. 146–155.

[62] Quemada, J.: *Compressed State Space Representation in LOTOS with the Interleaved Expansion.* In [44], pp. 19–35.

[63] Rauhamaa, M.: *A Comparative Study of Methods for Efficient Reachability Analysis.* Helsinki University of Technology, Digital Systems Laboratory Report A 14, Espoo 1990, 61 p.

[64] Reisig, W.: *Petri Nets: An Introduction.* EATCS Monographs on Theoretical Computer Science 4, Springer-Verlag, Berlin 1985, 161 p.

[65] Reisig, W.: *Place/Transition Systems.* In [8], pp. 117–141.

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

[67] Schmidt, K. and Starke, P.H.: *An Algorithm to Compute the Symmetries of Petri Nets.* Petri Net Newsletter 40 (1991), pp. 25–30.

[68] Starke, P.H.: *Reachability Analysis of Petri Nets Using Symmetries.* Systems Analysis – Modelling – Simulation 8 (1991) 4/5, pp. 293–303.

[69] Tiusanen, M.: *Static Analysis of Ada Tasking Programs: Models and Algorithms.* PhD Thesis, University of Illinois at Chicago, Chicago IL 1993, 161 p.

[70] Valmari, A.: *Error Detection by Reduced Reachability graph generation.* Proceedings of the 9th European Workshop on Application and Theory of Petri Nets, Venice, June 1988, pp. 95–112.

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

[72] Valmari, A.: *Some Polynomial Space Complete Concurrency Problems.* Tampere University of Technology, Software Systems Laboratory Report 4, Tampere 1988, 34 p.

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

[74] Valmari, A.: *Eliminating Redundant Interleavings during Concurrent Program Verification.* Proceedings of Parallel Architectures and Languages Europe '89 Vol. 2. Lecture Notes in Computer Science 366, Springer-Verlag, Berlin 1989, pp. 89–103.

[75] Valmari, A.: *Stubborn Sets for Reduced State Space Generation.* Rozenberg, G. (Ed.), Advances in Petri Nets 1990. Lecture Notes in Computer Science 483, Springer-Verlag, Berlin 1991, pp. 491–515.

[76] Valmari, A.: *A Stubborn Attack on State Explosion.* Formal Methods in System Design 1 (1992) 4, pp. 297–322.

[77] Valmari, A.: *Compositional State Space Generation.* University of Helsinki, Department of Computer Science, Report A-1991-5, Helsinki 1991, 30 p.

[78] Valmari, A.: *Stubborn Sets of Coloured Petri Nets.* Proceedings of the 12th International Conference on Application and Theory of Petri Nets, Gjern, Denmark, June 1991, pp. 102–121.

[79] Valmari, A.: *Alleviating State Explosion during Verification of Behavioural Equivalence.* University of Helsinki, Department of Computer Science, Report A-1992-4, Helsinki 1992, 57 p.

[80] Valmari, A.: *On-the-Fly Verification with Stubborn Sets.* In [16], pp. 397–408.

[81] Valmari, A. and Tienari, M.: *An Improved Failures Equivalence for Finite-State Systems with a Reduction Algorithm.* In [44], pp. 3–18.

[82] Valmari, A. and Tiusanen, M.: *A Graph Model for Efficient Reachability Analysis of Description Languages.* Proceedings of the 8th European Workshop on Application and Theory of Petri Nets, Zaragoza, June 1987, pp. 349–366.

[83] Varpaaniemi, K. and Rauhamaa, M.: *The Stubborn Set Method in Practice.* In [42], pp. 389–393.

[84] Vautherin, J.: *Parallel Systems Specifications with Coloured Petri Nets and Algebraic Specifications.* Rozenberg, G. (Ed.), Advances in Petri Nets 1987. Lecture Notes in Computer Science 266, Springer-Verlag, Berlin 1987, pp. 293–308.

[85] Wheeler, G.R., Valmari, A., and Billington, J.: *Baby TORAS Eats Philosophers but Thinks About Solitaire.* Proceedings of the 5th Australian Software Engineering Conference, Sydney, May 1990, pp. 283–288.

[86] Wolper, P. and Godefroid, P.: *Partial-Order Methods for Temporal Verification.* In [4], pp. 233–246.

[87] Wolper, P., Godefroid, P., and Pirottin.: *A Tutorial on Partial-Order Methods for the Verification of Concurrent Systems.* Tutorial material of the 5th International Conference on Computer-Aided Verification, Elounda, Greece, June/July 1993, 85 p.

[88] Wolper, P. and Leroy, D.: *Reliable Hashing without Collision Detection.* In [16], pp. 59–70.

[89] Yoneda, T., Shibayama, A., Schlingloff, B.-H., and Clarke, E.M.: *Efficient Verification of Parallel Real-Time Systems.* In [16], pp. 321–332.