Kim G. Larsen
Peter Niebert (Eds.)

Formal Modeling and Analysis of Timed Systems

LNCS 2791

First International Workshop, FORMATS 2003
Marseille, France, September 2003
Revised Papers

Springer

Kim G. Larsen Peter Niebert (Eds.)

Formal Modeling and Analysis of Timed Systems

First International Workshop, FORMATS 2003
Marseille, France, September 6-7, 2003
Revised Papers

Springer

Volume Editors

Kim G. Larsen
Aalborg University, Department of Computer Science
Fr. Bajersvej 7E, 9220 Aalborg East, Denmark
E-mail: kgl@cs.auc.dk

Peter Niebert
Université de Provence, Laboratoire d'Informatique Fondamentale, CMI
39, Rue Joliot-Curie, 13453 Marseille Cedex 13, France
E-mail: peter.niebert@lif.univ-mrs.fr

Library of Congress Control Number: 2004103614

CR Subject Classification (1998): F.3, D.2, D.3, C.3

ISSN 0302-9743
ISBN 3-540-21671-5 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN: 10931844 06/3142 5 4 3 2 1 0

Preface

This volume contains the proceedings of the 1st International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS 2003) held as a satellite event of CONCUR 2003 in Marseille, France, September 6–7, 2003. FORMATS and CONCUR were hosted by the Université de Provence and the Laboratoire d'Informatique Fondamentale de Marseille (LIF).

Timed Systems. Traditionally, timing aspects of systems from a variety of computer science domains are treated independently in separate scientific disciplines: People who are interested in semantics, verification or performance analysis are working on models such as timed automata, timed Petri nets or max-plus algebra. Electrical engineers have to consider propagation delays in their circuits and designers of embedded controllers have to take into account the time it takes for a controller to compute its reaction after sampling the environment.

While indeed the timing-related questions in these separate disciplines have their particularities (e.g., worst-case analysis vs. average-case optimization), there is a growing awareness of the difficult problems common to all of them, suggesting the interdisciplinary study of timed systems: the unifying theme underlying all these apparently different domains is that they treat systems whose behavior depends upon combinations of logical and temporal constraints, i.e., constraints on the distance between the occurrences of two events.

FORMATS is a new workshop aiming to be a major annual event dedicated to the study of timed systems, uniting three independently started workshop series related to the topic: MTCS (held as a satellite event of CONCUR 2000–2002), RT-TOOLS (held as a satellite event of CONCUR 2001 and FLoC 2002) and TPTS (at ETAPS 2002), with a total in 2002 of around 100 individual participants.

Of the 36 papers submitted to the first FORMATS workshop, 19 were selected for presentation and publication. In addition to these contributions, invited talks were given by Evgeny Asarin (VERIMAG, France), Paul Pettersen (University of Uppsala, Sweden) and Reinhard Wilhelm (University of Saarbrücken, Germany).

We would like to thank the Program Committee members and the referees who assisted us in the evaluation of the submitted papers. Also, many thanks go to the other members of the local Organization Committee, in particular Silvano Dal Zilio, Rémi Morin, Sarah Zennou, Pedro D'Argenio and all the members of the MOVE team. We gratefully acknowledge the particular support from the European project IST-2001-35304 AMETIST (Advanced MEthods in TImed SysTems), as well as the sponsors of CONCUR: Conseil Général des Bouches du Rhônes, Région Provence-Alpes-Côte d'Azur, Ville de Marseille, Université de Provence, Université de la Méditerranée and Laboratoire d'Informatique Fondamentale de Marseille.

July 2003 Kim G. Larsen and Peter Niebert

Organization

Steering Committee

Rajeev Alur (USA)
Kim G. Larsen (Denmark)
Walter Vogler (Germany)
Flavio Corradini (Italy)
Oded Maler (France)
Wang Yi (Sweden)

Program Committee

Rajeev Alur (USA)
Alan Burns (United Kingdom)
Jordi Cortadella (Spain)
Thomas A. Henzinger (USA)
Marta Kwiatkowska (United Kingdom)
Kim G. Larsen (Denmark, co-chair)
Oded Maler (France)
Ernst-Rüdiger Olderog (Germany)
Wojciech Penczek (Poland)
P.S. Thiagarajan (Singapore)
Frits Vaandrager (Netherlands)
Wang Yi (Sweden)
Jos Baeten (Netherlands)
Flavio Corradini (Italy)
Pedro D'Argenio (Argentina)
Joost-Pieter Katoen (Netherlands)
François Laroussinie (France)
Claude LePape (France)
Peter Niebert (France, co-chair)
Paritosh K. Pandya (India)
Olaf Stursberg (Germany)
Stavros Tripakis (France)
Walter Vogler (Germany)

Referees

Christel Baier
Maria Rita Di Berardini
Elmar Bihler
Behzad Bordbar
Patricia Bouyer
Nadia Busi
Franck Cassez
Robert Clariso
Henning Dierks
Emmanuel Fleury
Wan Fokkink
Norman Gethin
Stephen Gilmore
Michael Harrison
Martijn Hendriks
Holger Hermanns
Jens Peter Holmegaard
Tomas Krilavicius
Rom Langerak
Kamal Lodaya
Natalia Lopez
Gerald Luettgen
Nicolas Markey
Kees Middelburg
Sjouke Mauw
Emmanuel Fleury
Ileana Ober
Denis Oddoux
David Parker
Marco A. Peña
Agata Polrola
Franck Pommereau
Olivier Henri Roux
Andreas Schäfer
Fernando Schapachnik
Luis Sierra
Arne Skou
Ana Sokolova
Jeremy Sproston
Maciej Szreter
Yaroslav Usenko
Mark van der Zwaag
Tim Willemse
Bozena Wozna
Sergio Yovine
Marcelo Zanconi
Andrzej Zbrzezny
Sarah Zennou

Table of Contents

Timed Automata and Timed Languages Challenges and Open Problems
Eugene Asarin 1

Towards Efficient Partition Refinement for Checking Reachability in Timed Automata
Agata Półrola, Wojciech Penczek, and Maciej Szreter 2

Checking ACTL* Properties of Discrete Timed Automata via Bounded Model Checking
Bożena Woźna and Andrzej Zbrzezny 18

Removing Irrelevant Atomic Formulas for Checking Timed Automata Efficiently
Jianhua Zhao, Xuandong Li, Tao Zheng, and Guoliang Zheng 34

Adding Symmetry Reduction to UPPAAL
Martijn Hendriks, Gerd Behrmann, Kim Larsen, Peter Niebert, and Frits Vaandrager 46

TIMES: A Tool for Schedulability Analysis and Code Generation of Real-Time Systems
Tobias Amnell, Elena Fersman, Leonid Mokrushin, Paul Pettersson, and Wang Yi 60

Optimization of Timed Automata Models Using Mixed-Integer Programming
Sebastian Panek, Olaf Stursberg, and Sebastian Engell 73

Discrete-Time Rewards Model-Checked
Suzana Andova, Holger Hermanns, and Joost-Pieter Katoen 88

Performance Analysis of Probabilistic Timed Automata Using Digital Clocks
Marta Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston 105

An Interval-Based Algebra for Restricted Event Detection
Jan Carlson and Björn Lisper 121

PARS: A Process Algebra with Resources and Schedulers
MohammadReza Mousavi, Michel Reniers, Twan Basten, and Michel Chaudron 134

Formal Semantics of Hybrid Chi
R.R.H. Schiffelers, D.A. van Beek, K.L. Man, M.A. Reniers, and J.E. Rooda 151

Run-Time Guarantees for Real-Time Systems
Reinhard Wilhelm 166

A Nonarchimedian Discretization for Timed Languages
Cătălin Dima 168

Folk Theorems on the Determinization and Minimization of Timed Automata
Stavros Tripakis 182

Control Synthesis for a Smart Card Personalization System Using Symbolic Model Checking
Biniam Gebremichael and Frits Vaandrager 189

On Timing Analysis of Combinational Circuits
Ramzi Ben Salah, Marius Bozga, and Oded Maler 204

Analysis of Real Time Operating System Based Applications
Libor Waszniowski and Zdenek Hanzalek 219

Time-Optimal Test Cases for Real-Time Systems
Anders Hessel, Kim G. Larsen, Brian Nielsen, Paul Pettersson, and Arne Skou 234

Using Zone Graph Method for Computing the State Space of a Time Petri Net
Guillaume Gardey, Olivier H. Roux, and Olivier F. Roux 246

Causal Time Calculus
Franck Pommereau 260

ELSE: A New Symbolic State Generator for Timed Automata
Sarah Zennou, Manuel Yguel, and Peter Niebert 273

Author Index 281

Timed Automata and Timed Languages Challenges and Open Problems*

Eugene Asarin

VERIMAG, Centre Equation
2 ave de Vignate, 38610 Gières
France
Eugene.Asarin@imag.fr

Abstract. The first years of research in the area of timed systems were marked by a spectacular progress, but also by many natural and important problems left behind without solutions. Some of those are really hard, some have been completely overlooked, some are known only to small groups of researchers but have never been really attacked by the community.
The aim of this talk is to present several open problems and research directions in the domain of timed systems which seem important to the author. In particular we will consider variants of timed automata, theory of timed languages, timed games etc.

* Partially supported by the European community project IST-2001-35304 AMETIST

K.G. Larsen and P. Niebert (Eds.): FORMATS 2003, LNCS 2791, p. 1, 2004.

Towards Efficient Partition Refinement for Checking Reachability in Timed Automata*

Agata Półrola[1], Wojciech Penczek[2,3], and Maciej Szreter[2]

[1] Faculty of Mathematics
University of Lodz
Banacha 22, 90-238 Lodz, Poland
polrola@math.uni.lodz.pl
[2] Instituite of Computer Science
PAS
Ordona 21, 01-237 Warsaw, Poland
{penczek,mszreter}@ipipan.waw.pl
[3] Institute of Informatics
Podlasie Academy
Sienkiewicza 51, 08-110 Siedlce, Poland

Abstract. The paper presents a new method for building abstract models for Timed Automata, enabling on-the-fly reachability analysis. Our *pseudo-simulating models*, generated by a modified partitioning algorithm, are in many cases much smaller than *forward-reachability graphs* commonly applied for this kind of verification. A theoretical description of the method is supported by some preliminary experimental results.

1 Introduction

Model checking is an approach commonly applied for automated verification of *reachability properties*. Given a system and a property p, reachability model checking consists in an exploration of the (reachable) state space of the system, testing whether there exists a state where p holds. The main problem of this approach is caused by the size of the state space, which in many cases, in particular for timed systems, can be very large (even infinite). One of the solutions to this problem consists in applying finite *abstract models* of systems, preserving reachability properties. To this aim, *forward-reachability graphs* are most commonly used [6, 8, 14]. Reachability analysis on these models is usually performed *on-the-fly*, while generating a model, i.e., given a property p, newly obtained states of the model are examined, and the generation of the model is finished as soon as a state satisfying p is found [6]. An alternative solution are *symbolic methods*, one of which, very intensively investigated recently, consists in exploiting SAT-based Bounded Model Checking (BMC) [3, 21]. In the BMC approach, satisfiability of a formula encoding reachability of a state satisfying p

* Partly supported by the State Committee for Scientific Research under the grant No. 8T11C 01419

K.G. Larsen and P. Niebert (Eds.): FORMATS 2003, LNCS 2791, pp. 2–17, 2004.

is tested, using a symbolic path of a bounded length encoding the unfolding of a transition relation. Since the length of this path affects dramatically the size of its propositional encoding, the BMC methods are mainly applicable for proving reachability, but can become ineffective when no state satisfying p can be found (see the discussion in the section on experimental results). Therefore, verification methods based on building (small) abstract models of systems still have a practical importance, and developing efficient algorithms for generating such models remains an important subject of research.

Our paper presents a new method for generating abstract models of Timed Automata using a modified minimization (partitioning) algorithm [5]. The very first motivation for our approach has been taken from [20], where the authors claim that minimal bisimulating models (b-models, for short) for Timed Automata could often be smaller than the corresponding forward-reachability ones (fr-models, for short). Since *simulating* (*s-*) *models* [17] are usually smaller than minimal b-models, they could be used instead of the latter. However, it is clear that there should exist abstract models preserving reachability properties that are even smaller than the minimal s-models, as the latter preserve the whole language of ACTL. To define these models we relax the requirement on the transition relation of the s-models, formulated for all the predecessors of each state, such that it applies to one of them only, and call the new class of models pseudo-simulating ones (ps-models, for short). The models can be generated using a modification of the partitioning algorithm for s-models [10]. Moreover, the method can be used in an on-the-fly manner for reachability verification.

The rest of the paper is organised as follows: Section 2 presents the related work. In Section 3, we introduce Timed Automata and their concrete and abstract models usually considered in the literature. Then, in Sections 4 - 6 we provide a definition, an algorithm, and an implementation of ps-models for Timed Automata. Sections 7 and 8 contain experimental results and final remarks.

2 Related Work

Different aspects of the reachability analysis for Timed Automata have been usually studied on fr-models [8, 14, 16]. In [8], some abstractions allowing to reduce their sizes are proposed, while in [14], data structures for effective verification are shown. Alternative methods of reachability verification consist in exploiting SAT-solvers [3, 21], BDDs (a solution for closed automata shown in [4]), untimed histories of states and a bisimulation relation [13], or partitioning to obtain pseudo-b-models [19]. Partitioning-based reachability analysis was studied also for other kinds of systems [7, 15]. Moreover, the paper [12] presents various reachability-preserving equivalence relations. We provide a comparison with models generated by these relations in the full version of this work [18].

Minimization algorithms for b-models were introduced in [5, 15]. The first of them was applied to s-models in [10]. Implementations for Timed Automata and b-models can be found in [1, 2, 20, 22], and for s-models - in [11]. The

paper [20] contains some examples showing that b-models can be smaller than the corresponding fr- ones.

3 Timed Automata

Let $\mathbb{R}$ ($\mathbb{R}_+$) denote the set of (non-negative) reals, and $\mathbb{N}$ - the set of natural numbers. Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a finite set of variables, called *clocks*. A *valuation* on $\mathcal{X}$ is a n-tuple $v = (v_1, \ldots, v_n) \in \mathbb{R}_+^n$, where v_i is the value of the clock x_i in v. For a valuation v and $\delta \in \mathbb{R}$, $v + \delta$ denotes the valuation v' s.t. for all $x_i \in \mathcal{X}$, $v'_i = v_i + \delta$. For a valuation v and a subset of clocks $X \subseteq \mathcal{X}$, $v[X := 0]$ denotes the valuation v' such that for all $x_i \in X$, $v'_i = 0$ and for all $x_i \in \mathcal{X} \setminus X$, $v'_i = v_i$. By an *atomic constraint* for $\mathcal{X}$ we mean an expression of the form $x_i \sim c$ or $x_i - x_j \sim c$, where $x_i, x_j \in \mathcal{X}$, $\sim \in \{\leq, <, >, \geq\}$ and $c \in \mathbb{N}$. A valuation v *satisfies* an atomic constraint $x_i \sim c$ ($x_i - x_j \sim c$) if $v_i \sim c$ ($v_i - v_j \sim c$, respectively). A (*time*) *zone* of $\mathcal{X}$ is a convex polyhedron in $\mathbb{R}_+^n$ defined by a finite set of atomic constraints, i.e., the set of all the valuations satisfying all these constraints. The set of all the time zones of $\mathcal{X}$ is denoted by $Z(n)$.

Definition 1. *A* timed automaton $\mathcal{A}$ *is a tuple* $(\Sigma, S, \mathcal{X}, s^0, E, \mathcal{I})$, *where* Σ *is a finite set of* actions, $\mathcal{X} = \{x_1, \ldots, x_n\}$ *is a finite set of clocks,* $E \subseteq S \times \Sigma \times Z(n) \times 2^{\mathcal{X}} \times S$ *is a* transition relation. *Each element* e *of* E *is denoted by* $s \xrightarrow{a,z,Y} s'$, *which represents a transition from location* s *to* s', *performing an action* a, *with the set* $Y \subseteq \mathcal{X}$ *of clocks to be reset, and with a zone* z *defining the enabling condition for* e. *The function* $\mathcal{I} : S \longrightarrow Z(n)$, *called a* location invariant, *assigns to each location a zone defining the conditions under which* $\mathcal{A}$ *can be in this location.*

A *concrete state* of $\mathcal{A}$ is a pair $q = (s, v)$, where $s \in S$ and $v \in \mathbb{R}_+^n$ is a valuation such that $v \in \mathcal{I}(s)$. The set of all the concrete states is denoted by Q. The initial state of $\mathcal{A}$ is $i = 1, \ldots, n$. The states of $\mathcal{A}$ can change as a result of passing some time or performing an action as follows: the automaton can change from (s, v) to (s', v') on $e \in E$ (denoted by $(s, v) \xrightarrow{e}_d (s', v')$) iff $e : s \xrightarrow{a,z,Y} s'$, $v \in z$, and $v' = v[Y := 0] \in \mathcal{I}(s')$; and can change from (s, v) to (s', v') by passing some time $\delta \in \mathbb{R}_+$ (denoted by $(s, v) \xrightarrow{\delta}_d (s', v')$) iff $s = s'$ and $v' = v + \delta \in \mathcal{I}(s)$. The structure $F_d = (Q, q^0, \rightarrow_d)$ is the *concrete dense state space* of $\mathcal{A}$.

Besides the relation $\rightarrow_d$ defined above, other kinds of transition relations can also be introduced. For our purposes, we define the *concrete* (*discrete*) *successor relation* $\rightarrow \in Q \times E \times Q$ as follows: for $q, q' \in Q$ and $e \in E$, let $q \xrightarrow{e} q'$ denote that q' is obtained from q by passing some time, performing the transition $e \in E$, and then passing some time again. Formally, $q \xrightarrow{e} q'$ iff $(\exists q_1, q_2 \in Q)(\exists \delta_1, \delta_2 \in \mathbb{R}_+)$ $q \xrightarrow{\delta_1}_d q_1 \xrightarrow{e}_d q_2 \xrightarrow{\delta_2}_d q'$. The state q' is called a *successor* of q, whereas the structure $F_c = (Q, q^0, \rightarrow)$ is called the *concrete* (*discrete*) *state space* of $\mathcal{A}$.

Let $q \in Q$. A q-run of $\mathcal{A}$ is a finite sequence of concrete states $q_0 \xrightarrow{e_0} q_1 \xrightarrow{e_1} q_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{n-1}} q_n$, where $q_0 = q$ and $e_i \in E$ for each $i < n$. A state $q' \in Q$ is

reachable if there exists a q^0-run and $i \in \mathbb{N}$ such that $q' = q_i$. The set of all the reachable states of $\mathcal{A}$ will be denoted by $Reach_{\mathcal{A}}$.

3.1 Models for Timed Automata

Let PV be a set of propositional variables, and let $V_c : Q \to 2^{PV}$ be a valuation function, which assigns the same propositions to the states with the same location, i.e., $V_c((s,v)) = V_c((s',v'))$ for all $s = s'$.

Definition 2. *Let $F_c = (Q, q^0, \to)$ be the concrete (discrete) state space of a timed automaton $\mathcal{A}$. A structure $M_c = (F_c, V_c)$ is called a* concrete (discrete) model *of $\mathcal{A}$.*

Since concrete state spaces (and therefore concrete models) of Timed Automata are usually infinite, they cannot be directly applied to model checking. Therefore, in order to reduce their sizes we define finite abstractions, preserving properties to be verified. The idea is to combine into classes (sets) the concrete states that are indistinguishable w.r.t. these properties.

Definition 3. *Let $M_c = (F_c, V_c)$ be a concrete model for $\mathcal{A}$. A structure $M = (G, V)$, where $G = (W, w_0, \to)$ is a directed, rooted, edge-labelled* [1] *graph with a node set W, $w_0 \in W$ is the initial node, and $V : W \to 2^{PV}$ is a valuation function, is called an* abstract (discrete) model *for $\mathcal{A}$ if the following conditions are satisfied:*

- *each node $w \in W$ is a set of states of Q and $q^0 \in w_0$;*
- *for each $w \in W$ and $q \in w$ we have $V_c(q) = V(w)$;*
- $(\forall w_1, w_2 \in Reach(W))(\forall e \in E)\ w_1 \xrightarrow{e} w_2$ *iff* $(\exists q_1 \in w_1)(\exists q_2 \in w_2)\ q_1 \xrightarrow{e} q_2$, *where* $Reach(W) = \{w \in W \mid w \cap Reach_{\mathcal{A}} \neq \emptyset\}$.

The graph G is called an abstract state space *of $\mathcal{A}$, whereas its nodes are called* abstract states. *The abstract model M is* complete *iff* $(\forall q \in Q)(\exists w \in W)\ q \in w$.

In what follows, we consider complete abstract models only.

In the literature, abstract models generated for a *dense semantics* (i.e., derived from the concrete state space F_d) are usually considered. One of them are *surjective models.* Below, we provide their definition adapted for the discrete case:

Definition 4. *A model $M = (G, V)$ for $\mathcal{A}$, where $G = (W, w_0, \to)$, is called* surjective *iff*
$(\forall w_1, w_2 \in Reach(W))(\forall e \in E)$ *if* $w_1 \xrightarrow{e} w_2$ *then* $(\forall q_2 \in w_2)(\exists q_1 \in w_1)\ q_1 \xrightarrow{e} q_2$.

An example of surjective models are forward reachability (fr-) models, commonly applied for reachability verification [6, 8, 14]. Reachability analysis on these models is usually performed *on-the-fly*, together with their generation [6]

[1] The edges are labelled with the names of transitions in E.

(notice that in the worst case the whole model must be generated). The models can be further improved by applying various abstractions [8, 14].

Another class of abstract models considered in the literature are *bisimulating* (*b-*) *models*. These models are usually generated for the dense semantics [1, 20], but again their definition can be easily adapted also for the discrete one:

Definition 5. *A model* $M = (G, V)$ *for* $\mathcal{A}$, *where* $G = (W, w_0, \rightarrow)$, *is* bisimulating *iff*
$(\forall w_1, w_2 \in Reach(W))(\forall e \in E)$ *if* $w_1 \xrightarrow{e} w_2$ *then* $(\forall q_1 \in w_1)(\exists q_2 \in w_2)\ q_1 \xrightarrow{e} q_2$.

Moreover, in [17], the following *simulating* (*s-*) *models* were introduced:

Definition 6. *A model* $M = (G, V)$ *for* $\mathcal{A}$, *where* $G = (W, w_0, \rightarrow)$, *is* simulating *iff for each* $w \in W$ *there exists a non-empty* $w^{cor} \subseteq w$ *such that* $q^0 \in w_0^{cor}$ *and* $(\forall w_1, w_2 \in Reach(W))(\forall e \in \mathrm{E})$ *if* $w_1 \xrightarrow{e} w_2$ *then* $(\forall q_1 \in w_1^{cor})(\exists q_2 \in w_2^{cor})\ q_1 \xrightarrow{e} q_2$.

Both b- and s- models preserve reachability properties.

3.2 Zones and Regions

Finite abstract models built for Timed Automata use *regions* as states.

Definition 7. *Given a timed automaton* $\mathcal{A}$, *let* $s \in S$, *and* $Z \in Z(n)$. *A* region $R \subseteq S \times \mathbb{R}_+^n$ *is a set of states* $R = \{(s, v) \mid v \in Z\}$, *denoted by* (s, Z). *The region* $(s, \emptyset)$ *is identified with the empty region.*

Let $v, v' \in \mathbb{R}_+^n$, $Z, Z' \in Z(n)$, and $R, R' \in S \times Z(n)$. We define the following operations on zones and regions:

- $v \leq v'$ iff $\exists \delta \in \mathbb{R}_+$ such that $v' = v + \delta$;
- $Z \setminus Z'$ is a set of disjoint zones s.t. $\{Z'\} \cup (Z \setminus Z')$ is a partition of Z;
- $R \setminus R' = \{(s, Z'') \mid Z'' \in Z \setminus Z'\}$ for regions $R = (s, Z)$ and $R' = (s, Z')$;
- $Z[Y := 0] = \{v[Y := 0] \mid v \in Z\}$; $\quad [Y := 0]Z = \{v \mid v[Y := 0] \in Z\}$;
- $Z \nearrow := \{v' \in \mathbb{R}^n \mid (\exists v \in Z)\ v \leq v'\}$; $\quad Z \swarrow := \{v' \in \mathbb{R}^n \mid (\exists v \in Z)\ v' \leq v\}$.

Notice that the operations $\cap$, $\nearrow$, $\swarrow$, $Z[Y := 0]$ and $[Y := 0]Z$ preserve zones. These results together with the implementation of $Z \setminus Z'$ can be found in [1, 20].

4 Pseudo-simulating Models

In [20], the authors claim that minimal b-models, generated for the dense semantics, are often smaller than the corresponding fr- ones. Since s-models are usually smaller than the former, they could be better for reachability verification. However, in order to test reachability even more effectively, we introduce *pseudo-simulating* (*ps-*) *models* (which are never bigger than s- ones), and provide an algorithm for an on-the-fly reachability verification. The idea behind the definition of ps-models consists in relaxing the requirement on the transition

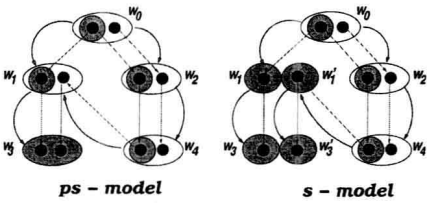

Fig. 1. A ps- and s-model generated for the same case

relation of the s-models, formulated for all the predecessors of each state (see Def. 6), such that it applies to one of them only. The selected predecessor needs to be reachable from the beginning state in the minimal number of steps.

Before we give the definition, we need some auxiliary notions. For two nodes w, w' of G, let $w \rightarrow w'$ denote that there exists $e \in E$ s.t. $w \xrightarrow{e} w'$. A *path* π in G is a finite sequence of nodes and edges of the form $\pi = w_1 \xrightarrow{e_1} w_2 \xrightarrow{e_2} \ldots \xrightarrow{e_{k-1}} w_k$, with $e_i \in E$ for all $i < k$ (labels on the edges can be then omitted). We say that π is from w_1 to w_k. A path is of length k if it contains k edges. For a node $w \in W$, the *depth* of w, denoted by $dpt(w)$, is the length of a shortest path from w_0 to w in G if there is such, otherwise, the depth of w is assumed to be infinite.

Definition 8. *A model $M = (G, V)$ for $\mathcal{A}$, where $G = (W, w_0, \rightarrow)$, is* pseudo-simulating *iff for each $w \in W$ there exists a non-empty $w^{cor} \subseteq w$ such that $q^0 \in w_0^{cor}$, and*
$(\forall w_1, w_2 \in Reach(W))(\forall e \in \mathrm{E})$ *if $w_1 \xrightarrow{e} w_2$, then there exists $w \in Reach(W)$ and $h \in \mathrm{E}$ such that $w \xrightarrow{h} w_2$ and $dpt(w)$ is minimal in the set $\{dpt(w') \mid w' \xrightarrow{h'} w_2$, for some $h' \in E\}$, and (*) $(\forall q_1 \in w^{cor})$ $(\exists q_2 \in w_2^{cor})$ $q_1 \xrightarrow{h} q_2$.*

The following example shows a difference between s- and ps- models:

Example 1. Fig. 1 presents a ps- and s-model generated for the same case. The *cor*s of the classes are coloured; circles and straight lines are used for drawing the concrete model, while ellipses and arcs - for abstract ones. In the ps-model, the state of w_4^{cor} does not need to have successors in w_1^{cor}. This, however, is required in the s-model, which results in creating two additional nodes w_1' and w_3'.

Let $M = (G, V)$, where $G = (W, w_0, \rightarrow)$, be a ps-model for $\mathcal{A}$. A run $\rho = q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} q_n$ of $\mathcal{A}$ is said to be *inscribed* in a path $\pi = w_0 \xrightarrow{e_1} w_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} w_n$ in G, if $q_i \in w_i$ for all $i = 0, \ldots, n$.

Denote all the edges $w \xrightarrow{e} w_2$ in G satisfying the condition (*) of Def. 8 by $w \overset{e}{\Rightarrow} w'$. Moreover, let $w \Rightarrow w'$ denote that there exists $e \in E$ s.t. $w \overset{e}{\Rightarrow} w'$. Next, we characterise ps-models:

Theorem 1. *The following conditions hold:*

a) Each q^0-run of $\mathcal{A}$ is inscribed in a path of G,
b) For each $w \in Reach(W)$, there is $n \in \mathbb{N}$ and $\pi = w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n$ in G s.t. $w = w_n$,

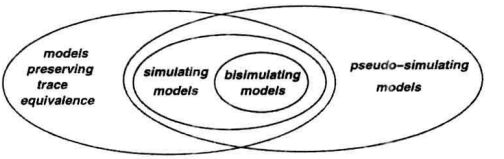

Fig. 2. Relations between various kinds of models

c) For each path $\pi = w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n$, *there exists a* q^0*-run* $\rho = q^0 \xrightarrow{e_0} q_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{n-1}} q_n$ *of* $\mathcal{A}$ *inscribed in* π *and such that* $q_i \in w_i^{cor}$ *for each* $i \leq n$.

A proof can be found in [18]. It it easy to see from the above theorem that the ps-models preserve reachability.

Fig. 2 shows the relations between ps-models and some other well-known classes of models considered in the literature. A proof and an extended comparison, including also other classes of models, can be found in the full version of this paper [18].

5 A Minimization Algorithm for Ps-Models

Ps-models can be generated using a modification of the well-known *minimization (partitioning) algorithm* [5]. In order to give the algorithm, we introduce the following notions:

By a *partition* $\Pi \subseteq 2^Q$ of the set of concrete states Q of $\mathcal{A}$ we mean a set of disjoint *classes* $X \subseteq Q$ the union of which equals Q. For a given partition Π of Q, $X, Y \in \Pi$ and $e \in E$ we introduce the functions:

- $pre_e(X,Y) = \{x \in X \mid \exists y \in Y : x \xrightarrow{e} y\}$;
- $post_e(X,Y) = \{y \in Y \mid \exists x \in X : x \xrightarrow{e} y\}$.

In order to generate ps-models, instead of a partition of Q, we use a *d-cor-partition* $\Pi \subseteq 2^Q \times 2^Q \times (\mathbb{N} \cup \{\infty\})$, defined as a set of triples of the form $\widehat{X} = (X, X^{cor}, dpt(X))$, where $\Pi|_1$ (i.e., the projection of Π on the first component) is a partition of Q, and $X^{cor} \subseteq X$. By $q \in \widehat{X}$ we mean that $q \in X$. Define $\widehat{X} \xrightarrow{e} \widehat{Y}$ iff $X \xrightarrow{e} Y$. Moreover, we introduce

- $Pre^e_\Pi(\widehat{X}) = \{\widehat{Y} \in \Pi \mid pre_e(Y,X) \neq \emptyset\}$, $Pre_\Pi(\widehat{X}) = \bigcup_{e \in E} Pre^e_\Pi(\widehat{X})$,
- $Post^e_\Pi(\widehat{X}) = \{\widehat{Y} \in \Pi \mid post_e(X,Y) \neq \emptyset\}$, $Post_\Pi(\widehat{X}) = \bigcup_{e \in E} Post^e_\Pi(\widehat{X})$.

A class $\widehat{X}$ is *reachable* if there is a concrete state $q \in X$ which is reachable.

Below, we introduce the notion of *ps*-unstability. Intuitively, a class is *ps*-unstable w.r.t. its successor $\widehat{Y}$ in Π if there is no predecessor of $\widehat{Y}$ with a minimal depth such that its *cor* contains only states with successors in Y^{cor} (see also Fig. 3).

Definition 9. *Let* Π *be a given d-cor-partition, and* $\widehat{X}, \widehat{Y} \in \Pi$.

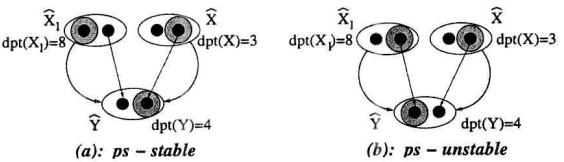

Fig. 3. *Ps*-stability and *ps*-unstability

- *The class* $\widehat{X}$ *is ps*-unstable *w.r.t.* $\widehat{Y}$ *iff for some* $e \in E$ *we have* $pre_e(X, Y) \neq \emptyset$*, and for all* $h \in E$ *and all* $\widehat{X_1} \in \Pi$ *such that* $\widehat{X_1} \xrightarrow{h} \widehat{Y}$ *and* $dpt(X_1)$ *is minimal in* $\{dpt(X_1') \mid \widehat{X_1'} \in Pre_\Pi(\widehat{Y})\}$ *we have* $pre_h(X_1^{cor}, Y^{cor}) \neq X_1^{cor}$*;*
- Π *is ps*-stable *iff* $(G \mid_1, V \mid_1)$*, where* $G \mid_1 = (\Pi \mid_1, [q^0], \rightarrow)$*, is a ps-model with* X^{cor} *and* $dpt(X)$ *satisfying Def. 8 w.r.t.* X *for each* $\widehat{X} \in \Pi$*.*

Example 2. Fig. 3 illustrates the notions of *ps*-stability and *ps*-unstability. Consider classes $\widehat{X}, \widehat{X_1}, \widehat{Y}$ of a partition Π with the components *dpt* as shown in the figure. In the part (a), both the classes $\widehat{X}$ and $\widehat{X_1}$ are *ps*-stable w.r.t. $\widehat{Y}$, since all the states of X^{cor} have successors in Y^{cor}, and $\widehat{X}$ is the predecessor of $\widehat{Y}$ with the minimal depth. In contrary, in (b) both the classes are *ps*-unstable w.r.t. $\widehat{Y}$, since its predecessor $\widehat{X}$ does not satisfy the required condition.

The minimization algorithm for ps-models is a modification of the algorithm for s-models [10]. It starts from an initial *d-cor*-partition Π_0, in which the component *dpt* of the class containing q^0 is equal to 0 and its *cor* is the singleton $\{q^0\}$, whereas for all the other classes $\widehat{X} \in \Pi_0$, $dpt(X) = \infty$ and $X^{cor} = X$. Then, it constructs a minimal model $M^{ps}_{min} = (G^{ps}_{min}, V)$, where $G^{ps}_{min} = (\Pi^{st}, ([q^0], \{q^0\}, 0), \rightarrow)$, Π^{st} is the reachable part of a *ps*-stable partition Π obtained by a refinement of Π_0, $\Pi \mid_1$ is *compatible* with $\Pi_0 \mid_1$ (i.e., each class of $\Pi_0 \mid_1$ is a union of classes of $\Pi \mid_1$), and $q^0 \in ([q^0], \{q^0\}, 0)$. The algorithm is parameterised by a non-deterministic function $Split(\widehat{X}, \Pi)$, defined for the classes $\widehat{X} \in \Pi$ with $dpt(X) \neq \infty$ (the explanation for considering these classes only will be given later). The function refines Π by choosing a class $\widehat{Y} \in \Pi$ w.r.t. which $\widehat{X}$ is *ps*-unstable, and then splitting either $\widehat{X}$, or a class $\widehat{X_1} \in \Pi$ s.t. $dpt(X_1)$ is minimal in the set $\{dpt(X_1') \mid \widehat{X_1'} \in Pre_\Pi(\widehat{Y})\}$, in order to make $\widehat{X}$ *ps*-stable w.r.t. $\widehat{Y}$. Before defining the above function, we introduce another function $dpt_\Pi(X)$, defined for $X \in \Pi \mid_1$, which is used for computing the component $dpt(X)$ when a new class $\widehat{X}$ is created. The function returns a value which is a possible depth of $\widehat{X}$, determined by the analysis of the classes $\widehat{Y} \in \Pi$ for which there is $e \in E$ s.t. $pre_e(Y, X) \neq \emptyset$ (notice that the components *dpt* of the classes in a given step of the algorithm can differ from their depths in the model obtained when the algorithm terminates). More precisely, $dpt_\Pi([q^0]) = 0$, $dpt_\Pi(X) = 1 + min\{dpt(V) \mid \widehat{V} \in \Pi \ \wedge \ pre_e(V, X) \neq \emptyset$ for some $e \in E\}$ if there exists $\widehat{V} \in \Pi$ and $e \in E$ such that $pre_e(V, X) \neq \emptyset$ and $dpt(V) \neq \infty$, and $dpt_\Pi(X) = \infty$, otherwise. For $\widehat{X}, \widehat{Y} \in \Pi$ s.t. $pre_e(X, Y) \neq \emptyset$ and $pre_e(X^{cor}, Y^{cor}) \neq X^{cor}$ for some $e \in E$,

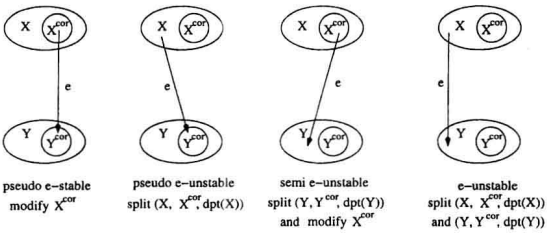

Fig. 4. The four cases of the function Sp

we define also an auxiliary function $Sp(\widehat{X}, \widehat{Y}, e, \Pi)$, which splits $\widehat{X}$ w.r.t. $\widehat{Y}$ as follows (see also Fig. 4):

1. $Sp(\widehat{X}, \widehat{Y}, e, \Pi) = \{(X, pre_e(X^{cor}, Y^{cor}), dpt_\Pi(X))\}$
   if $\widehat{X}$ is *pseudo e-stable* w.r.t. $(Y, Y^{cor}, dpt(Y))$, i.e., $pre_e(X^{cor}, Y^{cor}) \neq \emptyset$;
2. $Sp(\widehat{X}, \widehat{Y}, e, \Pi) =$
   $\{(X \setminus X^{cor}, pre_e(X, Y^{cor}), dpt_\Pi(X \setminus X^{cor})), (X^{cor}, X^{cor}, dpt_\Pi(X^{cor}))\}$ if $\widehat{X}$ is *pseudo e-unstable* w.r.t. $\widehat{Y}$, i.e., $pre_e(X^{cor}, Y^{cor}) = \emptyset \wedge pre_e(X, Y^{cor}) \neq \emptyset$;
3. $Sp(\widehat{X}, \widehat{Y}, e, \Pi) = \{(X, pre_e(X^{cor}, Y), dpt_\Pi(X)), (Y^{cor}, Y^{cor}, dpt_\Pi(Y^{cor})),$
   $(Y \setminus Y^{cor}, Y \setminus Y^{cor}, dpt_\Pi(Y \setminus Y^{cor}))\}$ if $\widehat{X}$ is *semi e-unstable* w.r.t. $\widehat{Y}$, i.e.,
   $pre_e(X^{cor}, Y^{cor}) = pre_e(X, Y^{cor}) = \emptyset \wedge pre_e(X^{cor}, Y) \neq \emptyset$;
4. $Sp(\widehat{X}, \widehat{Y}, e, \Pi) = \{(pre_e(X, Y), pre_e(X, Y), dpt_\Pi(pre_e(X, Y))),$
   $(X \setminus pre_e(X, Y), X^{cor}, dpt_\Pi(X \setminus pre_e(X, Y))), (Y^{cor}, Y^{cor}, dpt_\Pi(Y^{cor})), (Y \setminus$
   $Y^{cor}, Y \setminus Y^{cor}, dpt_\Pi(Y \setminus Y^{cor}))\}$ if $\widehat{X}$ is *e-unstable* w.r.t. $\widehat{Y}$,
   i.e., $pre_e(X^{cor}, Y^{cor}) = pre_e(X, Y^{cor}) = pre_e(X^{cor}, Y) = \emptyset$.

Then, we define

- $Split(\widehat{X}, \Pi) = \{\widehat{X}\}$ if $\widehat{X}$ is *ps*-stable w.r.t. all $\widehat{Y}$ in Π.

Otherwise, a class $\widehat{Y}$ and a transition $e \in E$ are chosen, for which $pre_e(X, Y) \neq \emptyset$ and $\widehat{X}$ is *ps*-unstable w.r.t. $\widehat{Y}$, and then

a) if $dpt(Y) \geq dpt(X) + 1$, then $Split(\widehat{X}, \Pi) = Sp(\widehat{X}, \widehat{Y}, e, \Pi)$;
b) if $dpt(Y) < dpt(X) + 1$, then we choose a class $\widehat{X_1}$ s.t. for some $h \in E$ we have $pre_h(X_1, Y) \neq \emptyset$, $pre_h(X_1^{cor}, Y) \neq X_1^{cor}$ and $dpt(X_1) = min\{dpt(X_1') \mid \widehat{X_1'} \in Pre_\Pi(\widehat{Y})\}$, and $Split(\widehat{X}, \Pi) = Sp(\widehat{X_1}, \widehat{Y}, h, \Pi)$.

Intuitively, if $\widehat{X}$ is *ps*-unstable w.r.t. $\widehat{Y}$ and from the analysis of Π of a given step we can assume that in the model obtained when the algorithm terminates $\widehat{X}$ will be the predecessor of $\widehat{Y}$ of the minimal depth, then we apply to these classes the appropriate case of the function Sp. Otherwise, i.e., if $dpt(Y)$ indicates that $\widehat{Y}$ has another predecessor with a depth smaller than $dpt(X)$, we apply the function Sp to $\widehat{Y}$ and to its predecessor of a smallest value of dpt (see also Fig. 5).