Basic Research in Computer Science

BRICS RS-01-5 Hune et al.: Linear Parametric Model Checking of Timed Automata

Linear Parametric Model Checking of Timed Automata

Thomas S. Hune
Judi Romijn
Mariëlle Stoelinga
Frits W. Vaandrager

BRICS Report Series **RS-01-5**

ISSN 0909-0878 **January 2001**

**Copyright © 2001, Thomas S. Hune & Judi Romijn & Mariëlle Stoelinga & Frits W. Vaandrager.
BRICS, Department of Computer Science
University of Aarhus. All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

**See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/01/5/`

Linear Parametric Model Checking of Timed Automata*

Thomas Hune[1], Judi Romijn[2], Mariëlle Stoelinga[2], and Frits Vaandrager[2]

[1] BRICS***, University of Århus, Denmark
baris@brics.dk
[2] Computing Science Institute, University of Nijmegen
[judi,marielle,fvaan]@cs.kun.nl

Abstract. We present an extension of the model checker UPPAAL capable of synthesize linear parameter constraints for the correctness of parametric timed automata. The symbolic representation of the (parametric) state-space is shown to be correct. A second contribution of this paper is the identification of a subclass of parametric timed automata (L/U automata), for which the emptiness problem is decidable, contrary to the full class where it is know to be undecidable. Also we present a number of lemmas enabling the verification effort to be reduced for L/U automata in some cases. We illustrate our approach by deriving linear parameter constraints for a number of well-known case studies from the literature (exhibiting a flaw in a published paper).

1 Introduction

During the last decade, there has been enormous progress in the area of timed model checking. Tools such as UPPAAL[14], KRONOS [6], and PMC [15] are now routinely used for industrial case studies. A disadvantage of the traditional approaches is, however, that they can only be used to verify concrete timing properties: one has to provide the values of all timing parameters that occur in the system. For practical purposes, one is often interested in deriving the (symbolic) constraints on the parameters that ensure correctness. The process of manually finding and proving such results is very time consuming and error prone (we have discovered minor errors in the two examples we have been looking at). Therefore tool support for deriving the constraints *automatically* is very important.

* Research supported by Esprit Project 26270, Verification of Hybrid Systems (VHS), and by PROGRESS Project TES4199, Verification of Hard and Softly Timed Systems (HaaST). This work was initiated during a visit of the first author to the University of Nijmegen.

*** Basic Research in Computer Science (www.brics.dk), funded by the Danish National Research Foundation.

In this paper, we study a parameterized extension of timed automata, as well as a corresponding extension of the forward reachability algorithm for timed automata. We show the theoretical correctness of our approach, and its feasibility by application to non-trivial case studies. For this purpose, we have implemented a prototype extension of UPPAAL, an efficient real-time model checking tool [14]. The algorithm we propose and have implemented is a semi-decision algorithm which will not terminate in all cases. In [3] the problem of synthesizing values for parameters such that a property is satisfied, was shown to be undecidable, so this is the best we can hope for.

A second contribution of this paper is the identification of a subclass of parameterized timed automata, called *lower bound/upper bound (L/U) automata*, which appears to be sufficiently expressive from a practical perspective, while it also has nice theoretical properties. Most importantly, we show that the emptyness problem for parametric timed automata, shown to be undecidable in [3], is decidable for L/U automata. We also establish a number of lemmas which allow one to reduce the number of parameters when tackling specific verification questions for L/U automata. The application of these lemmas has already reduced the verification effort drastically in some of our experiments.

Related work Our attempt at automatic verification of parameterized real-time models is not the only one. Henzinger et al. aim at solving a more general problem with HYTECH [11], a tool for model checking hybrid automata, exploring the state-space either by partition refinement, or forward reachability. The tool has been applied successfully to relatively small examples such as a railway gate controller. Experience so far has shown that HYTECH cannot cope with larger examples, such as the ones considered in this paper.

Toetenel et al. [15] have made an extension of the PMC real-time model checking tool [5] called LPMC. LPMC is restricted to linear parameter constraints as is our approach, and uses the partition refinement method, like HYTECH. Other differences with our approach are that LPMC also allows for the comparison of non-clock variables to parameter constraints, and for more general specification properties (full TCTL with fairness assumptions). Since LPMC is a quite recent tool, not many applications have been presented yet. However, a model of the IEEE 1394 root contention protocol inspired by [17] has been successfully analyzed in [5].

A more general attempt than LPMC and our UPPAAL extension has been made by Annichini et al. [4]. They have constructed and implemented a method which allows non-linear parameter constraints, and uses heavier, third-party, machinery to solve the arising non-linear constraint comparisons. Independently, we have used the same data-structure (a direct extension of DBMs [9]) for the symbolic representation of the state space, as in [4]. For speeding up the exploration, a method for guessing the effect of control loops in the model is presented. It appears that this helps termination of the method, but it is unclear under what circumstances this technique can or cannot be used. The feasibility of this approach has been shown on a few rather small case studies. One of these is Fischer's protocol with two processes, for which the state space is constructed in about 3 minutes cpu time.

The remainder of this paper is organized as follows. Section 2 introduces the notion of parametric timed automata. Section 3 gives the symbolic semantics, which is the basis for our model checking algorithm, presented in Section 3.5. Section 4 is an intermezzo that states some helpful lemmas and decidability results on an interesting subclass. Finally, Section 5 reports on experiments with our tool.

2 Parametric Timed Automata

2.1 Parameters and Constraints

Throughout this paper, we will be working with a fixed set of *parameters* $P = \{p_1, \ldots, p_n\}$.

Definition 1 (Constraints). *A* linear expression *e is either an expression of the form $t_1p_1+\cdots+t_np_n+t_0$, where $t_0, \ldots, t_n \in \mathsf{Z}$, or ∞. We write E to denote the set of all linear expressions. A* constraint *is an inequality of the form $e \sim e'$, with e, e' linear expressions and $\sim \in \{<, \leq, >, \geq\}$. The* negation *of constraint c, notation $\neg c$, is obtained by replacing relation signs $<$, $\leq$, $>$, $\geq$ by $\geq$, $>$, $\leq$, $<$, respectively. A* (parameter) valuation *is a function $v : P \to \mathsf{R}^{\geq 0}$ assigning a nonnegative real value to each parameter. There is a one-to-one correspondence between valuations and points in $(\mathsf{R}^{\geq 0})^n$. In fact we often identify a valuation v with the point $(v(p_1), \ldots, v(p_n)) \in (\mathsf{R}^{\geq 0})^n$.*

If e is a linear expression and v is a valuation, then $e[v]$ denotes the expression obtained by replacing each parameter p in e with $v(p)$. Likewise, we define $c[v]$ for c a constraint. Valuation v satisfies *constraint*

c, notation $v \models c$*, if* $c[v]$ *evaluates to true. The* semantics *of a constraint c, notation* $[\![c]\!]$*, is the set of valuations (points in* $(\mathsf{R}^{\geq 0})^n$*) that satisfy c. A finite set of constraints* C *is called a* constraint set*. A valuation* satisfies *a constraint set if it satisfies each constraint in the set. The* semantics *of a constraint set* C *is given by* $[\![C]\!] := \bigcap_{c \in C} [\![c]\!]$*. We write* $\top$ *to denote any constraint set with* $[\![\top]\!] = (\mathsf{R}^{\geq 0})^n$*, for instance the empty set. We use* $\bot$ *to denote any constraint set with* $[\![\bot]\!] = \emptyset$*, for instance the constraint set* $\{c, \neg c\}$*, for some arbitrary c.*

Constraint c covers *constraint set C, notation* $C \models c$*, iff* $[\![C]\!] \subseteq [\![c]\!]$*. Constraint set C* is split by *constraint c iff neither* $C \models c$ *nor* $C \models \neg c$*.*

During the analysis questions arise of the kind: given a constraint set C and a constraint c, does c hold, i.e., does constraint c cover C? There are three possible answers to this, *yes*, *no*, and *split*. A split occurs when c holds for some valuations in the semantics of C and $\neg c$ holds for some other valuations. We will not discuss methods for answering such questions: in our implementation we use an oracle to compute the following function.

Definition 2 (Oracle).

$$\mathcal{O}(c, C) = \begin{cases} \text{yes} & \textit{if } C \models c \\ \text{no} & \textit{if } C \models \neg c \\ \text{split} & \textit{otherwise} \end{cases}$$

Observe that using the oracle, we can easily decide semantic inclusion between constraint sets: $[\![C]\!] \subseteq [\![C']\!]$ iff $\forall c' \in C' : \mathcal{O}(c', C) = \text{yes}$. The oracle that we use is a linear programming (LP) solver that was kindly provided to us by the authors of [5], who built it for their LPMC model checking tool. This LP solver is geared to perform well on small, simple sets of constraints rather than large, complicated ones.

2.2 Parametric Timed Automata

Throughout this paper, we assume a fixed set of clocks $X = \{x_0, \ldots, x_m\}$ and a fixed set of actions $A = \{a_1, \ldots, a_k\}$. The special clock x_0, which is called the *zero clock*, always has the value 0 (and hence does not increase with time).

A *simple guard* is an expression f of the form $x_i - x_j \prec e$, where x_i, x_j are clocks, $\prec \in \{<, \leq\}$, and e is a linear expression. We say that f is *proper* if $i \neq j$. We define a *guard* to be a (finite) conjunction of simple guards. We let g range over guards and write G to denote the set of guards. A

clock valuation is a function $w : X \to \mathsf{R}^{\geq 0}$ assigning a nonnegative real value to each clock, such that $w(x_0) = 0$. We will identify a clock valuation w with the point $(w(x_0), \ldots, w(x_m)) \in (\mathsf{R}^{\geq 0})^{m+1}$. Let g be a guard, v a parameter valuation, and w a clock valuation. Then $g[v, w]$ denotes the expression obtained by replacing each parameter p with $v(p)$, and each clock x with $w(x)$. A pair (v, w) of a parameter valuation and a clock valuation *satisfies* a guard g, notation $(v, w) \models g$, if $g[v, w]$ evaluates to true. The *semantics* of a guard g, notation $[\![g]\!]$, is the set of pairs (v, w) such that $(v, w) \models g$.

A *reset* is an expression of the form, $x_i := b$ where $i \neq 0$ and $b \in \mathsf{N}$. A *reset set* is a set of resets containing at most one reset for each clock. The set of reset sets is denoted by R.

We now define an extension of timed automata [2, 20] called parametric timed automata. Similar models have been presented in [3–5].

Definition 3 (PTA). *A* parametric timed automaton (PTA) *over set of clocks X, set of actions A, and set of parameters P, is a quadruple $\mathcal{A} = (Q, q_0, \to, I)$, where Q is a finite set of* locations*, $q_0 \in Q$ is the* initial location*, $\to \subseteq Q \times A \times G \times R \times Q$ is a finite* transition relation*, and function $I : Q \to G$ assigns an* invariant *to each location. We abbreviate a $(q, a, g, r, q') \in \to$ consisting of a source location, an action, a guard, a reset set, and a target location as $q \xrightarrow{a,g,r} q'$. For a simple guard $x_i - x_j \prec e$ to be used in an invariant it must be the case that $x_j = x_0$, that is, the simple guard represents an upper bound on a clock.*

Example 1. A parametric timed automaton with clocks x, y and parameters p, q can be seen in Fig. 1. The initial state is $S0$ which has invariant $x \leq p$, and the transition from the initial location to $S1$ has guard $y \geq q$ and reset set $x := 0$. There are no actions on the transitions. Initially the transition from $S0$ to $S1$ is only enabled if $p \leq q$, otherwise the system will be deadlocked.

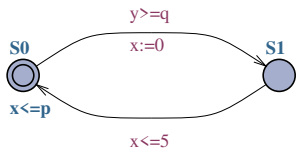

Fig. 1. A parametric timed automaton

To define the semantics of PTAs, we require two auxiliary operations on clock valuations. For clock valuation w and nonnegative real number d, $w+d$ is the clock valuation that adds to each clock (except x_0) a delay d. For clock valuation w and reset set r, $w[r]$ is the clock valuation that resets clocks according to r.

$$(w+d)(x) = \begin{cases} 0 & \text{if } x = x_0 \\ w(x) + d & \text{otherwise} \end{cases} \qquad (w[r])(x) = \begin{cases} b & \text{if } x := b \in r \\ w(x) & \text{otherwise.} \end{cases}$$

Definition 4 (LTS). *A* labeled transition system (LTS) *over a set of symbols Σ is a triple $\mathcal{L} = (S, S_0, \rightarrow)$, with S a set of* states*, $S_0 \subseteq S$ a set of* initial states*, and $\rightarrow \subseteq S \times \Sigma \times S$ a* transition relation*. We write $s \xrightarrow{a} s'$ for $(s, a, s') \in \rightarrow$. A* run *of $\mathcal{L}$ is a finite alternating sequence $s_0 a_1 s_1 a_2 \cdots s_n$ of states $s_i \in S$ and symbols $a_i \in \Sigma$ such that $s_0 \in S_0$ and, for all $i < n$, $s_i \xrightarrow{a_{i+1}} s_{i+1}$. A state is* reachable *if it is the last state of some run.*

Definition 5 (Concrete semantics). *Let $\mathcal{A} = (Q, q_0, \rightarrow, I)$ be a PTA and v be a parameter valuation. The* concrete semantics of $\mathcal{A}$ under v*, notation $[\![\mathcal{A}]\!]_v$, is the labeled transition system (LTS) $(S, S_0, \rightarrow)$ over $A \cup \mathsf{R}^{\geq 0}$ where*

$$S = \{(q, w) \in Q \times (X \to \mathsf{R}^{\geq 0}) \mid w(x_0) = 0 \wedge (v, w) \models I(q)\},$$
$$S_0 = \{(q, w) \in S \mid q = q_0 \wedge w = \lambda x.0\},$$

and transition predicate $\rightarrow$ is specified by the following two rules, for all (q, w), $(q', w') \in S$, $d \geq 0$ and $a \in A$,

- $(q, w) \xrightarrow{d} (q', w')$ *if $q = q'$ and $w' = w + d$.*
- $(q, w) \xrightarrow{a} (q', w')$ *if $\exists g, r : q \xrightarrow{a,g,r} q' \wedge (v, w) \models g \wedge w' = w[r]$.*

Note that the LTS $[\![\mathcal{A}]\!]_v$ has at most one initial state (at most, since we require that all states satisfy the location invariants).

2.3 The Problem

In its current version, UPPAAL is able to check for reachability properties, in particular whether certain combinations of locations and constrains on clock variables are reachable from the initial configuration. Our parameterized extension of UPPAAL handles exactly the same properties. However, rather than just telling whether a property holds or not, our tool looks for constraints on the parameters which ensure that the property holds.

Definition 6 (Properties). *The sets of* system properties *and* state formulas *are defined by, respectively,*

$$\psi ::= \forall\Box\phi \mid \exists\Diamond\phi \qquad\qquad \phi ::= x - y \prec b \mid q \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi$$

where $x, y \in X$, $b \in \mathsf{N}$ *and* $q \in Q$. *Let* $\mathcal{A}$ *be a PTA,* v *a parameter valuation,* s *a state of* $[\![\mathcal{A}]\!]_v$, *and* ϕ *a state formula. We write* $s \models \phi$ *if* ϕ *holds in state* s, *we write* $[\![\mathcal{A}]\!]_v \models \forall\Box\phi$ *if* ϕ *holds in all reachable states of* $[\![\mathcal{A}]\!]_v$, *and we write* $[\![\mathcal{A}]\!]_v \models \exists\Diamond\phi$ *if* ϕ *holds for some reachable state of* $[\![\mathcal{A}]\!]_v$.

The problem that we address in this paper can now be stated as follows: *Given a parametric timed automaton* $\mathcal{A}$ *and a system property* ψ, *compute the set of parameter valuations* v *for which* $[\![\mathcal{A}]\!]_v \models \psi$.

Remark 1. Timed automata [2, 20] arise as a special case of PTAs for which the set P of parameters is empty. If $\mathcal{A}$ is a PTA and v is a parameter valuation, then the structure $\mathcal{A}[v]$ that is obtained by replacing all linear expressions e that occur in $\mathcal{A}$ by $e[v]$ is a timed automaton.[1] It is easy to see that in general $[\![\mathcal{A}]\!]_v = [\![\mathcal{A}[v]]\!]$. Since the reachability problem for timed automata is decidable [2], this implies that, for any $\mathcal{A}$, integer valued v and ψ, $[\![\mathcal{A}]\!]_v \models \psi$ is decidable.

2.4 Example: Fischer's Mutual Exclusion Protocol

Figure 2 shows a PTA model of Fischer's mutual exclusion protocol [13]. The purpose of this protocol is to guarantee mutually exclusive access to a critical section among competing processes P_1, $P_2, \ldots$ P_n. In this protocol, a shared variable lock is used for communication between the processes, with each process P_i running the following algorithm.

lock := 0;
REPEAT
  while lock $\neq 0$ **do skip**;
  lock := i;
  delay
UNTIL lock = i;
critical section;
lock := 0

[1] Strictly speaking, $\mathcal{A}[v]$ is only a timed automaton if v assigns an integer to each parameter.

The correctness of this algorithm crucially depends on the timing of the operations. The key idea for the correctness is that any process P_i that sets $\mathsf{lock} := i$ is made to wait long enough before checking $\mathsf{lock} = i$ to ensure that any other process P_j that tested $\mathsf{lock} = 0$, before P_i set lock to its index, has already set lock to its index j, when P_i finally checks $\mathsf{lock} = i$.

Assume that read/write access to the global variable (in the operations $\mathsf{lock} = i$ and $\mathsf{lock} := 0$) takes between min_rw and max_rw time units and assume that the delay operation (including the timed needed for the the assignment $\mathsf{lock} := i$) takes between min_delay and max_delay time units. If we assume the basic constraints $0 \leq min_rw < max_rw \wedge 0 \leq min_delay < max_delay$, then mutual exclusion is guaranteed if and only if $max_rw \leq min_delay$.

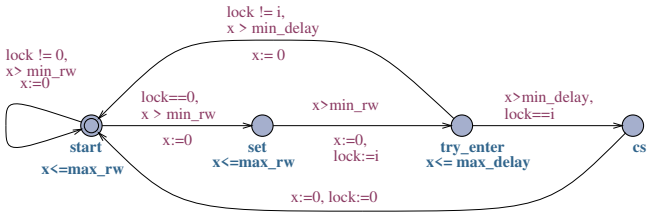

Fig. 2. A PTA model of Fischer's mutual exclusion protocol

Now consider the PTA in Fig. 2. (Several different models of this protocol exist [1, 3, 16, 19]; our model is closest to the one in [16].) It consists of four locations *start* (which is initial), *set*, *try_enter* and *cs*; four parameters, *min_rw*, *max_rw*, *min_delay* and *max_delay*; one clock x and a shared variable *lock*. By convention, x and *lock* are initially 0. Note that the process can remain in the locations *start* and *set* for at least *min_rw* and strictly less than *max_rw* time units. Similarly, the process can remain in *try_enter* for a time in the interval $[min_delay, max_delay)$.

The shared variable, which is not a part of the definition of PTAs, is syntactic sugar which allows for an efficient encoding of the protocol as a PTA. Also the notion of parallel composition for PTAs is standard. We refer the reader to [14] for their definitions.

3 Symbolic State Exploration

Our aim is to use basically the same algorithm for parametric timed model checking as for timed model checking. We represent sets of states symbolically in a similar way and support the same operations used for timed model checking. In the nonparametrized case, sets of states can be efficiently represented using matrices [9]. Similarly, in this paper we represent sets of states symbolically as *(constrained) parametric difference-bound matrices*.

Basically the same approach was followed in [4], although not worked out in detail. New in our presentation is the systematic use of structural operational semantics to deal with the nondeterministic computation that takes place in the parametrized case.

3.1 Parametric Difference-Bound Matrices

In the nonparametrized case, a difference-bound matrix is a $(m+1) \times (m+1)$ matrix whose entries are elements from $(\mathsf{Z} \cup \{\infty\}) \times \{0,1\}$. An entry $(c,1)$ for D_{ij} denotes a nonstrict bound $x_i - x_j \leq c$, whereas an entry $(c,0)$ denotes a strict bound $x_i - x_j < c$. Here, instead of using integers in the entries, we will use linear expressions over the parameters. Also, we find it convenient to view the matrix slightly more abstractly as a set of guards.

Definition 7 (PDBM). *A* parametric difference-bound matrix (PDBM) *is a set D which contains, for all $0 \leq i,j \leq m$, a simple guard D_{ij} of the form $x_i - x_j \prec_{ij} e_{ij}$. We require that, for all i, D_{ii} is of the form $x_i - x_i \leq 0$. Given a parameter valuation v, the* semantics *of D is defined by $[\![D]\!]_v = [\![\bigwedge_{i,j} D_{ij}]\!]_v$. We say that D is* satisfiable *for v if $[\![D]\!]_v$ is nonempty. If f is a proper guard of the form $x_i - x_j \prec e$ then we write $D[f]$ for the PDBM obtained from D by replacing D_{ij} by f. If i,j are indices then we write D^{ij} for the pair $(e_{ij}, \prec_{ij})$; we call D^{ij} a* bound *of D. Clearly, a PDBM is fully determined by its bounds.*

Definition 8 (Constrained PDBM). *A* constrained PDBM *is a pair (C,D) where C is a constraint set and D is a PDBM. The* semantics *of a constrained PDBM is defined by $[\![C,D]\!] = \{(v,w) \mid v \in [\![C]\!] \wedge w \in [\![D]\!]_v\}$.*

PDBMs with the tightest possible bounds are called *canonical*. To formalize this notion, we define an addition operation on linear expressions

by

$$(t_1 p_1 + \cdots + t_n p_n + t_0) + (t'_1 p_1 + \cdots + t'_n p_n + t'_0) \\ = (t_1 + t'_1) p_1 + \cdots + (t_n + t'_n) p_n + (t_0 + t'_0).$$

Also, we view Boolean connectives as operations on relation symbols $\leq$ and $<$ by identifying $\leq$ with 1 and $<$ with 0. Thus we have, for instance, $(\leq \wedge \leq) = \leq$, $(\leq \wedge <) = <$, $\neg \leq = <$, and $(\leq \Longrightarrow <) = <$. Our definition of a canonical form of a constrained PDBM is essentially equivalent to the one for standard DBMs.

Definition 9 (Canonical Form). *A constrained PDBM (C, D) is* in canonical form *iff for all i, j, k, $C \models e_{ij}\ (\prec_{ij} \Longrightarrow \prec_{ik} \wedge \prec_{kj})\ e_{ik} + e_{kj}$.*

The proof of the following technical lemma is immediate from the definitions.

Lemma 1.

1. *If $v \models e \prec e'$ and $v \models e' \prec e''$ then $v \models e\ (\prec \wedge \prec')\ e''$.*
2. *If $(v, w) \models x - y \prec e$ and $v \models e \prec' e'$ then $(v, w) \models x - y\ (\prec \wedge \prec')\ e'$.*
3. *If $v \models e\ (\prec \wedge \prec')\ e'$ then $v \models e \prec e'$.*
4. *If $(v, w) \models x - y\ (\prec \wedge \prec')\ e$ then $(v, w) \models x - y \prec e$.*
5. *If $(v, w) \models x - y \prec e$ and $(v, w) \models y - z \prec' e'$ then $(v, w) \models x - z\ (\prec \wedge \prec')\ e + e'$.*
6. *$v \models \neg(e \prec e')$ iff $v \models e'\ (\neg \prec)\ e$.*

The next important lemma, which basically carries over from the unparametrized case, states that canonicity of a constrained PDBM guarantees satisfiability. We recall the proof, since we will need the same argument later on in this section.

Lemma 2. *Suppose (C, D) is a constrained PDBM in canonical form and $v \in [\![C]\!]$. Then D is satisfiable for v.*

Proof. Inductively we will construct a valuation $(t_0, \ldots, t_i)$ for variables $(x_0, \ldots, x_i)$ such that all constraints D_{jk} for $0 \leq j, k \leq i$ are met.

To begin with, we set $t_0 = 0$. Then, trivially, $(v, x_0 \mapsto t_0) \models D_{00}$.

For the induction step, suppose that for some $i < n$ we have a valuation $(t_0, \ldots, t_i)$ for variables $(x_0, \ldots, x_i)$ such that all constraints D_{jk} for $0 \leq j, k \leq i$ are met. In order to extend this valuation to x_{i+1}, we have to

find a value t_{i+1} such that the following simple guards hold for valuation $(v, x_0 \mapsto t_0, \ldots, x_{i+1} \mapsto t_{i+1})$:

$$D_{i+1,0} \quad \cdots \quad D_{i+1,i} \quad D_{0,i+1} \quad \cdots \quad D_{i,i+1} \quad D_{i+1,i+1} \tag{1}$$

Here the first $i+1$ simple guards give upper bounds for t_{i+1}, the second $i+1$ simple guards give lower bounds for t_{i+1}, and the last simple guard is trivially met by any choice for t_{i+1}. We claim that each of the upper bounds is larger than each of the lower bounds. In particular, the minimum of the upper bounds is larger than the maximum of the lower bounds. This gives us a nonempty interval of possible values for t_{i+1} to choose from. Formally, we claim that, for all $0 \leq j, k < i+1$, the following formula holds for valuation $(v, x_0 \mapsto t_0, \ldots, x_i \mapsto t_i)$:

$$x_j - e_{j,i+1} \prec_{j,i+1} \wedge \prec_{i+1,k} x_k + e_{i+1,k} \tag{2}$$

To see why (2) holds, observe that by induction hypothesis $(v, x_0 \mapsto t_0, \ldots, x_i \mapsto t_i) \models$

$$x_j - x_k \prec_{jk} e_{jk} \tag{3}$$

Furthermore, since (C, D) is canonical,

$$e_{jk} \, (\prec_{jk} \Longrightarrow \prec_{j,i+1} \wedge \prec_{i+1,k}) \, e_{j,i+1} + e_{i+1,k} \tag{4}$$

Combination of (3) and (4), using Lemma 1(1), gives $(v, x_0 \mapsto t_0, \ldots, x_i \mapsto t_i) \models$

$$x_j - x_k \prec_{j,i+1} \wedge \prec_{i+1,k} e_{j,i+1} + e_{i+1,k}$$

which is equivalent to (2). This means that we can choose t_{i+1} in accordance with all the guards of (1), which completes the proof of the induction step and thereby of the lemma.

The following lemma essentially carries over from the unparametrized case too, see for instance [9]. As a direct consequence, semantic inclusion of constrained PDBMs is decidable for canonical PDBMs (using the oracle function).

Lemma 3. *Suppose $(C, D), (C', D')$ are constrained PDBMs and (C, D) is canonical. Then $[\![C, D]\!] \subseteq [\![C', D']\!] \Leftrightarrow ([\![C]\!] \subseteq [\![C']\!] \wedge \forall i, j : C \models e_{ij} (\prec_{ij} \Longrightarrow \prec'_{ij}) e'_{ij})$.*

3.2 Operations on PDBMs

Our algorithm requires basically four operations to be implemented on constrained PDBMs: adding guards, canonicalization, resetting clocks and computing time successors.

Adding Guards In the case of DBMs, adding a guard is a simple operation. It is implemented by taking the conjunction of a DBM and the guard (which is also viewed as a DBM). The conjunction operation just takes the pointwise minimum of the entries in both matrices. In the parametric case, adding a guard to a constrained PDBM may result in a set of constrained PDBMs. We define a relation $\overset{f}{\Leftarrow\!\shortmid}$ which relates a constrained PDBM and a guard to a collection of constrained PDBMs that satisfy this guard. For this we need an operation $\mathcal{C}$ that takes a PDBM and a simple guard, and produces a constraint stating that the bound imposed by the guard is larger than the corresponding bound in the PDBM, so let $D^{ij} = (e_{ij}, \prec_{ij})$ then

$$\mathcal{C}(D, x_i - x_j \prec e) = e_{ij}\ (\prec_{ij} \Longrightarrow \prec)\ e.$$

Relation $\overset{f}{\Leftarrow\!\shortmid}$ is defined as the smallest relation that satisfies the following rules:

$$(R1)\ \frac{\mathcal{O}(\mathcal{C}(D,f), C) = \text{yes}}{(C,D) \overset{f}{\Leftarrow\!\shortmid} (C,D)} \qquad (R2)\ \frac{\mathcal{O}(\mathcal{C}(D,f), C) = \text{no},\ f \text{ proper}}{(C,D) \overset{f}{\Leftarrow\!\shortmid} (C, D[f])}$$

$$(R3)\ \frac{\mathcal{O}(\mathcal{C}(D,f), C) = \text{split}}{(C,D) \overset{f}{\Leftarrow\!\shortmid} (C \cup \{\mathcal{C}(D,f)\}, D)} \qquad (R4)\ \frac{\mathcal{O}(\mathcal{C}(D,f), C) = \text{split},\ f \text{ proper}}{(C,D) \overset{f}{\Leftarrow\!\shortmid} (C \cup \{\neg\mathcal{C}(D,f)\}, D[f])}$$

$$(R5)\ \frac{(C,D) \overset{g}{\Leftarrow\!\shortmid} (C',D')\ ,\ (C'D') \overset{g'}{\Leftarrow\!\shortmid} (C'',D'')}{(C,D) \overset{g \wedge g'}{\Leftarrow\!\shortmid} (C'',D'')}$$

If the oracle replies "yes", then adding a simple guard will not change the constrained PDBM. If the answer is "no" then we tighten the bound in the PDBM according to the simple guard. With the answer "split" there are two possibilities and two pairs with updated constraint systems are returned. The side condition "f proper" in rules $R2$ and $R4$ ensures that the diagonal bounds in the PDBM always remain equal to $(0, \leq)$. If we update a bound in D then the semantics of the PDBM may become empty. The following lemma characterizes $\overset{f}{\Leftarrow\!\shortmid}$ semantically.

Lemma 4. $[\![C,D]\!] \cap [\![g]\!] \ = \ \bigcup\{[\![C',D']\!] \mid (C,D) \overset{g}{\Leftarrow\!\shortmid} (C',D')\}$.

Proof. "$\subseteq$". Assume $(v, w) \in [\![C, D]\!] \wedge (v, w) \models g$. By structural induction on g we prove that there exists a constrained PDBM (C', D') such that $(C, D) \overset{g}{\Leftarrow\!\!\shortmid} (C', D')$ and $(v, w) \in [\![C', D']\!]$.

For the induction basis, suppose g is of the form $x_i - x_j \prec e$. We consider four cases:

- $\mathcal{O}(\mathcal{C}(D, g), C) =$ yes. Let $C' = C$ and $D' = D$. Then trivially $(v, w) \in [\![C', D']\!]$ and, by rule $R1$, $(C, D) \overset{g}{\Leftarrow\!\!\shortmid} (C', D')$.
- $\mathcal{O}(\mathcal{C}(D, g), C) =$ no. By contradiction we prove that g is proper. Suppose g is not proper. Then, since $i = j$ and $v \models \neg e_{ij}(\prec_{ij} \Longrightarrow \prec)e$, $v \models \neg(0 \prec e)$. By Lemma 1(6), $v \models e\neg \prec 0$. But $(v, w) \models g$ implies $v \models 0 \prec e$. Hence, by Lemma 1(1), $v \models 0 < 0$, a contradiction. Let $C' = C$ and $D' = D[g]$. Then, by rule $R2$, $(C, D) \overset{g}{\Leftarrow\!\!\shortmid} (C', D')$. Since $(v, w) \in [\![C, D]\!]$ and $(v, w) \models g$, it follows that $(v, w) \in [\![C', D']\!]$.
- $\mathcal{O}(\mathcal{C}(D, g), C) =$ split and $v \models \mathcal{C}(D, g)$. Let $C' = C \cup \{\mathcal{C}(D, g)\}$ and $D' = D$. Then, by rule $R3$, $(C, D) \overset{g}{\Leftarrow\!\!\shortmid} (C', D')$. Moreover, by the assumptions, $(v, w) \in [\![C', D']\!]$.
- $\mathcal{O}(\mathcal{C}(D, g), C) =$ split and $v \models \neg\mathcal{C}(D, g)$. By contradiction we prove that g is proper. Suppose g is not proper. Then, since $v \models \neg\mathcal{C}(D, g)$, $v \models \neg(0 \prec e)$. By Lemma 1(6), $v \models e\neg \prec 0$. But $(v, w) \models g$ implies $v \models 0 \prec e$. Hence, by Lemma 1(1), $v \models 0 < 0$, a contradiction. Let $C' = C \cup \{\neg\mathcal{C}(D, g)\}$ and $D' = D[g]$. Then, by rule $R4$, $(C, D) \overset{g}{\Leftarrow\!\!\shortmid} (C', D')$. By the assumptions $(v, w) \in [\![C', D']\!]$.

For the induction step, suppose that g is of the form $g' \wedge g''$. Then $(v, w) \models g'$. By induction hypothesis, there exist C'', D'' such that $(C, D) \overset{g'}{\Leftarrow\!\!\shortmid} (C'', D'')$ and $(v, w) \in [\![C'', D'']\!]$. Since $(v, w) \models g''$, we can use the induction hypothesis once more to infer that there exist C', D' such that $(C'', D'') \overset{g''}{\Leftarrow\!\!\shortmid} (C', D')$ and $(v, w) \in [\![C', D']\!]$. Moreover, by rule $R5$, $(C, D) \overset{g}{\Leftarrow\!\!\shortmid} (C', D')$.

"$\supseteq$" Assume $(C, D) \overset{g}{\Leftarrow\!\!\shortmid} (C', D')$ and $(v, w) \in [\![C', D']\!]$. By induction on size of the derivation of $(C, D) \overset{g}{\Leftarrow\!\!\shortmid} (C', D')$, we establish $(v, w) \in [\![C, D]\!]$ and $(v, w) \models g$. There are five cases, depending on the last rule r used in the derivation of $(C, D) \overset{g}{\Leftarrow\!\!\shortmid} (C', D')$.

1. $r = R1$. Then $C = C'$, $D = D'$ and $C \models \mathcal{C}(D, g)$. Let g be of the form $x_i - x_j \prec e$. Hence $(v, w) \in [\![C, D]\!]$ and $v \models \mathcal{C}(D, g)$. By the first statement $(v, w) \models x_i - x_j \prec^D_{ij} e^D_{ij}$, and by the second statement $v \models e^D_{ij}\ (\prec^D_{ij} \Longrightarrow \prec)\ e$. Combination of these two observations, using parts (2) and (4) of Lemma 1 yields $(v, w) \models g$.

2. $r = R2$. Then $C = C'$, $D' = D[g]$ and $C \models \neg\mathcal{C}(D, g)$. Hence $(v, w) \models g$ and $v \models \neg\mathcal{C}(D, g)$. Let g be of the form $x_i - x_j \prec e$. By Lemma 1(6), $v \models e\ \neg(\prec_{ij}^D \Longrightarrow \prec)\ e_{ij}^D$. Using parts (2) and (4) of Lemma 1, combination of these two observations yields $(v, w) \models x_i - x_j \prec_{ij}^D e_{ij}^D$. Since trivially (v, w) is a model for all the other guards in D, $(v, w) \in [\![C, D]\!]$.

3. $r = R3$. Then $C' = C \cup \{\mathcal{C}(D, g)\}$ and $D' = D$. Let g be of the form $x_i - x_j \prec e$. We have $(v, w) \in [\![C, D]\!]$. This implies $(v, w) \models x_i - x_j \prec_{ij}^D e_{ij}^D$. We also have $v \models e_{ij}^D\ (\prec_{ij}^D \Longrightarrow \prec)\ e$. Combination of these two observations, using parts (2) and (4) of Lemma 1 yields $(v, w) \models g$.

4. $r = R4$. Then $C' = C \cup \{\neg\mathcal{C}(D, g)\}$ and $D' = D[g]$. We have $v \models \neg\mathcal{C}(D, g)$ and $(v, w) \models g$. Let g be of the form $x_i - x_j \prec e$. By Lemma 1(6), $v \models e\ \neg(\prec_{ij}^D \Longrightarrow \prec)\ e_{ij}^D$. Using parts (2) and (4) of Lemma 1 yields $(v, w) \models x_i - x_j \prec_{ij}^D e_{ij}^D$. Since trivially (v, w) is a model for all other guards in D, $(v, w) \in [\![C, D]\!]$.

5. $r = R5$. Then g is of the form $g' \wedge g''$ and there are C'', D'' such that $(C, D) \overset{g'}{\Leftarrow\!\shortmid} (C'', D'')$ and $(C'', D'') \overset{g''}{\Leftarrow\!\shortmid} (C', D')$. By induction hypothesis, $(v, w) \in [\![C'', D'']\!]$ and $(v, w) \models g''$. Again by induction hypothesis, $(v, w) \in [\![C, D]\!]$ and $(v, w) \models g'$. It follows that $(v, w) \models g$.

Canonicalization Each DBM can be brought into canonical form using classical algorithms for computing all-pairs shortest paths, for instance the Floyd-Warshall (FW) algorithm [7]. In the parametric case, we also apply this approach except that now we run FW *symbolically*.

The algorithm repeatedly compares the difference between two clocks to the difference obtained by looking at the difference when an intermediate clock is taken into account (the comparison used in Definition 9). In the symbolic case the result is, in general, a (possibly empty, finite) set of constrained PDBMs, rather than just a single matrix.

Below, we describe the computation steps of the symbolic FW algorithm in SOS style. Recall that the FW algorithm consists of three nested for-loops, for indices k, i and j, respectively. Correspondingly, in the SOS description of the symbolic version, we use configurations of the form (k, i, j, C, D), where (C, D) is a constrained PDBM and $k, i, j \in [0, m+1]$

record the values of indices. In the rules below, k, i, j range over $[0, m]$.

$$\frac{(C, D) \overset{x_i - x_j \prec_{ik} \wedge \prec_{kj} e_{ik} + e_{kj}}{\Leftarrow\!\!\!\!\dashv} (C', D')}{(k, i, j, C, D) \rightarrow_{FW} (k, i, j+1, C', D')}$$

$$(k, i, m+1, C, D) \rightarrow_{FW} (k, i+1, 0, C, D)$$

$$(k, m+1, 0, C, D) \rightarrow_{FW} (k+1, 0, 0, C, D)$$

We write $(C, D) \rightarrow_c (C', D')$ if there exists a sequence of $\rightarrow_{FW}$ steps leading from configuration $(0, 0, 0, C, D)$ to configuration $(m+1, 0, 0, C', D')$. In this case, we say that (C', D') is an *outcome* of the symbolic Floyd-Warshall algorithm on (C, D). It is easy to see that the set of all outcomes is finite and can be effectively computed. If the semantics of (C, D) is empty, then the set of outcomes is also empty. We write $(C, D) \overset{g}{\Leftarrow\!\!\!\!\dashv}_c (C', D')$ iff $(C, D) \overset{g}{\Leftarrow\!\!\!\!\dashv} (C'', D'') \rightarrow_c (C', D')$, for some C'', D''.

The following lemma says that if we run the symbolic Floyd-Warshall algorithm, the union of the semantics of the outcomes equals the semantics of the original constrained PDBM.

Lemma 5. $[\![C, D]\!] = \bigcup \{ [\![C', D']\!] \mid (C, D) \rightarrow_c (C', D') \}$.

Proof. By an inductive argument, using Lemma 4 and the fact that, for any valuation (v, w) in the semantics of (C, D),

$$\begin{aligned}
&(v, w) \models x_i - x_k \ \prec_{ik} \ e_{ik} \ \text{ and} \\
&(v, w) \models x_k - x_j \ \prec_{kj} \ e_{kj}, \ \text{ and therefore by Lemma 1(5)} \\
&(v, w) \models x_i - x_j \ \prec_{ik} \wedge \prec_{kj} \ e_{ik} + e_{kj}.
\end{aligned}$$

Lemma 6. *Each outcome of the symbolic Floyd-Warshall algorithm is a constrained PDBM in canonical form.*

Proof. As in [7].

Remark 2. Non-parametric DBMs can be canonicalized in $\mathcal{O}(n^3)$, where n is the number of clocks. In the parametric case, however, each operation of comparing the bound $D(x, x')$ to the weight of another path from x to x' may give rise to two new PDBMs if this comparison leads to a split. Then the two PDBMs must both be canonicalized to obtain all possible PDBMs with tightest bounds. Still, that part of these two PDBMs which was already canonical, does not need to be investigated again. So in the worst case, the cost of the algorithm becomes $\mathcal{O}(2^{n^3})$. In practice, it turns out that this is hardly ever the case.

Resetting Clocks A third operation on PDBMs that we need is resetting clocks. Since we do not allow parameters in reset sets, the reset operation on PDBMs is essentially the same as for DBMs, see [20]. If D is a PDBM and r is a singleton reset set $\{x_i := b\}$, then $D[r]$ is the PDBM obtained by (1) replacing each bound D^{ij}, for $j \neq i$, by $(e_{0j} + b, \prec_{0j})$; (2) replacing each bound D^{ji}, for $j \neq i$, by $(e_{j0} - b, \prec_{j0})$. We generalize this definition to arbitrary reset sets by

$$D[x_{i_1} := b_1, \ldots, x_{i_h} := b_h] = D[x_{i_1} := b_1] \ldots [x_{i_h} := b_h].$$

It easily follows from the definitions that resets preserves canonicity.

Lemma 7. *If (C, D) is canonical then $(C, D[r])$ is canonical as well.*

The following lemma characterizes the reset operation semantically.

Lemma 8. *Let (C, D) be a constrained PDBM in canonical form, $v \in [\![C]\!]$, and w a clock valuation. Then $w \in [\![D[r]]\!]_v$ iff $\exists w' \in [\![D]\!]_v : w = w'[r]$.*

Proof. We only prove the lemma for singleton resets. Using Lemma 7, the generalization to arbitrary resets is straightforward. Let $r = \{x_i := b\}$ and $D' = D[r]$.
"$\Leftarrow$" Suppose $w' \in [\![D]\!]_v$ and $w = w'[r]$. In order to prove $w \in [\![D']\!]_v$, we must show that $(v, w) \models D'_{kj}$, for all k and j. There are four cases:

1. $k \neq i \neq j$. Then $D'_{kj} = D_{kj}$. Since $(v, w') \models D_{kj}$ and w and w' agree on all clocks occurring in D_{kj}, $(v, w) \models D'_{kj}$.
2. $k = i = j$. Then $D'_{kj} = D_{kj}$. Since $(v, w') \models D_{ii}$, $0 \prec_{ii} e_{ii}[v]$. Hence $(v, w) \models D'_{kj}$.
3. $k \neq i = j$. Then $D'_{kj} = x_k - x_j \prec_{k0} e_{k0} - b$. Using that $(v, w') \models D_{k0}$, we derive $w(x_k) - w(x_j) = w'(x_k) - b \prec_{k0} e_{k0}[v] - b$. Hence $(v, w) \models D'_{kj}$.
4. $k = i \neq j$. Then $D'_{kj} = x_k - x_j \prec_{0j} e_{0j} + b$. Using that $(v, w') \models D_{0j}$, we derive $w(x_k) - w(x_j) = b - w'(x_j) \prec_{0j} e_{0j}[v] + b$. Hence $(v, w) \models D'_{kj}$.

"$\Rightarrow$" Suppose $w \in [\![D']\!]_v$. We have to prove that there exists a clock valuation $w' \in [\![D]\!]_v$ such that $w = w'[r]$. Clearly we need to choose w' in such a way that, for all $j \neq i$, $w'(x_j) = w(x_j)$. This means that, for any choice of $w'(x_i)$, for all $j \neq i \neq k$, $v, w' \models D_{jk}$. Using the same argument as in the proof of Lemma 2, we can find a value for $w'(x_i)$ such that also the remaining simple guards of D are satisfied.

Time Successors Finally, we need to transform PDBMs for the passage of time, notation $D\uparrow$. As in the DBMs case [9], this is done by setting the $x_i - x_0$ bounds to $(\infty, <)$, for each $i \neq 0$, and leaving all other bounds unchanged. We have the following lemma.

Lemma 9. *Suppose (C, D) is a constrained PDBM in canonical form, $v \in [\![C]\!]$, and w a clock valuation. Then $w \in [\![D\uparrow]\!]_v$ iff $\exists d \geq 0\ \exists w' \in [\![D]\!]_v : w' + d = w$.*

Proof. "$\Leftarrow$" Suppose $d \geq 0$, $w' \in [\![D]\!]_v$ and $w' + d = w$. We claim that $w \in [\![D\uparrow]\!]_v$. For this we must show that for each guard f of $D\uparrow$, $(v, w) \models f$. Let f be of the form $x_i - x_j \prec e$. We distinguish between three cases:

- $i \neq 0 \wedge j = 0$. In this case, by definition of $D\uparrow$, f is of the form $x_i - x_0 < \infty$, and so $(v, w) \models f$ trivially holds.
- $i = 0$. In this case f is also a constraint of D. Since $w' \in [\![D]\!]_v$ we have $(v, w') \models f$, and thus $-w'(x_j) \prec e[v]$. But since $d \geq 0$, this means that $-w(x_j) = -w'(x_i) - d \prec e[v]$ and therefore $(v, w) \models f$.
- $i \neq 0 \wedge j \neq 0$. In this case f is again a constraint of D. Since $w' \in [\![D]\!]_v$ we have $(v, w') \models f$, and therefore $w'(x_i) - w'(x_j) \prec e[v]$. But this means that $w'(x_i) - w'(x_j) = (w(x_i) - d) - (w(x_j) - d) \prec e[v]$ and therefore $(v, w) \models f$.

"$\Rightarrow$" Suppose $w \in [\![D\uparrow]\!]_v$. If $m = 0$ (i.e., there are no clocks) then $D\uparrow = D$ and the rhs of the implication trivially holds (take $w' = w$ and $d = 0$). So assume $m > 0$. For all indices i, j with $i \neq 0$ and $j \neq 0$, $(v, w) \models D_{ij}$. Hence $w(x_i) - w(x_j) \prec_{ij} e_{ij}[v]$. Thus, for any real number t, $w(x_i) - t - (w(x_j) - t) \prec_{ij} e_{ij}[v]$. But this means $(v, w - t) \models D_{ij}$. It remains to be shown that there exists a value d such that in valuation $(v, w - d)$ also the remaining guards D_{0i} and D_{i0} hold. Let

$$\begin{aligned} t_0 &= \max(0, w(x_1) - e_{10}[v], \ldots, w(x_n) - e_{n0}[v]) \\ t_1 &= \min(w(x_1) + e_{01}[v], \ldots, w(x_n) + e_{0n}[v]) \\ d &= (t_0 + t_1)/2 \\ w' &= w - d \end{aligned}$$

Intuitively, t_0 gives the least amount of time one has to go backwards in time from w to meet all upper bounds of D (modulo strictness), whereas t_1 gives the largest amount of time one can go backwards in time from w without violating any of the lower bounds of D (again modulo strictness). Value d sits right in the middle of these two. We claim that d and w' satisfy the required properties. For any i, since $(v, w) \models D_{0i}$, trivially

$$0 \prec_{0i} w(x_i) + e_{0i}[v] \tag{5}$$

Using that D is canonical we have, for any i, j,

$$e_{ji}[v] \ (\prec_{ji} \Longrightarrow \prec_{j0} \wedge \prec_{0i}) \ e_{j0}[v] + e_{0i}[v]$$

and, since $v, w \models D_{ji}$,

$$w(x_j) - w(x_i) \prec_{ji} e_{ji}[v].$$

Using these two observations we infer

$$w(x_j) - e_{j0}[v] \ (\prec_{ji} \Longrightarrow \prec_{j0} \wedge \prec_{0i}) \ w(x_j) - e_{ji}[v] + e_{0i}[v] \prec_{ji} w(x_i) + e_{0i}[v].$$

Hence

$$w(x_j) - e_{j0}[v] \prec_{j0} \wedge \prec_{0i} w(x_i) + e_{0i}[v] \tag{6}$$

By inequalities (5) and (6), each subterm of max in the definition of t_0 is dominated by each subterm of min in the definition of t_1. This implies $0 \leq t_0 \leq t_1$.

Now either $t_0 < t_1$ or $t_0 = t_1$. In the first case it easy to prove that in valuation (v, w) the guards D_{0i} and D_{i0} hold, for any i:

$$w'(x_i) = w(x_i) - d < w(x_i) - t_0 \leq w(x_i) - (w(x_i) - e_{i0}[v]) = e_{i0}[v]$$

and thus $w'(x_i) < e_{i0}[v]$ and $v, w' \models D_{i0}$. Also

$$-w'(x_i) = -w(x_i) + d < -w(x_i) + t_1 \leq -w(x_i) + (w(x_i) + e_{0i}[v]) = e_{0i}[v]$$

and so $-w'(x_i) < e_{0i}[v]$ and $v, w' \models D_{0i}$.

In the second case, fix an i. If $w(x_i) - e_{i0}[v] < t_0$ then

$$w'(x_i) = w(x_i) - d = w(x_i) - t_0 < w(x_i) - (w(x_i) - e_{i0}[v]) = e_{i0}[v]$$

and thus $w'(x_i) < e_{i0}[v]$ and $v, w' \models D_{i0}$. Otherwise, if $w(x_i) - e_{i0}[v] = t_0$ observe that by $t_0 = t_1$, inequality (6) and the fact that, $t_1 = w(x_j) + e_{0j}[v]$, for some j, $\prec_{i0} = \leq$. Since

$$w'(x_i) = w(x_i) - d \leq w(x_i) - t_0 \leq w(x_i) - (w(x_i) - e_{i0}[v]) \leq e_{i0}[v]$$

and thus $w'(x_i) \leq e_{i0}[v]$ this implies $v, w' \models D_{i0}$.

The proof that $v, w' \models D_{0i}$, for all i, in the case where $t_0 = t_1$ proceeds similarly.

3.3 Symbolic Semantics

With the four operations on PDBMs, we can describe the semantics of a parametric timed automaton symbolically.

Definition 10 (Symbolic semantics). *The* symbolic semantics *of PTA* $\mathcal{A} = (Q, q_0, \to, I)$ *is an LTS. The states are triples* (q, C, D) *with* q *a location from* Q *and* (C, D) *a constrained PDBM in canonical form. The set of initial states is*

$$\{(q_0, C, D) \mid (\top, \mathsf{E}\uparrow) \overset{I(q_0)}{\Leftarrow}_c (C, D)\},$$

where E *is the PDBM with* $\mathsf{E}^{ij} = (0, \leq)$*, for all* i, j*. The transitions are defined by the following rule:*

$$\frac{q \xrightarrow{a,g,r} q' \ , \ (C, D) \overset{g}{\Leftarrow}_c (C'', D'') \ , \ (C'', D''[r]\uparrow) \overset{I(q')}{\Leftarrow}_c (C', D')}{(q, C, D) \to (q', C', D')}.$$

Using Lemma 4 and Lemma 5, it follows by a simple inductive argument that if state (q, C, D) is reachable in the symbolic semantics and $(v, w) \in [\![C, D]\!]$ then $(v, w) \models I(q)$. It is also easy to see that the symbolic semantics of a PTA is a finitely branching transition system. It may have infinitely many reachable states though. Our search algorithm explores the symbolic semantics in an "intelligent" manner, and for instance stops whenever it reaches a state whose semantics is contained in the semantics of a state that has been encountered before. Despite this, our algorithm need not terminate.

Each run in the symbolic semantics can be simulated by a run in the concrete semantics.

Proposition 1. *For each parameter valuation* v *and clock valuation* w*, if there is a run in the symbolic semantics of* $\mathcal{A}$ *reaching state* (q, C, D)*, with* $(v, w) \in [\![C, D]\!]$*, then this run can be simulated by a run in the concrete semantics* $[\![\mathcal{A}]\!]_v$ *reaching state* (q, w)*.*

Proof. By induction on the number of transitions in the run.

As basis we consider a run with 0 transitions, i.e., a run that consists of a start state of the symbolic semantics. So this means that (q, C, D) is a start state.

Using the fact that $(v, w) \in [\![C, D]\!]$, the definition of start states, Lemma 5 and Lemma 4, we know that $q = q_0$, $(v, w) \models I(q_0)$ and $(v, w) \in [\![\top, \mathsf{E}\uparrow]\!]$. By Lemma 9, we get that there exists a $d \geq 0$ and $w' \in [\![\mathsf{E}]\!]_v$

such that $w' + d = w$. Since $(v, w) \models I(q_0)$ and invariants in a PTA only give upper bounds on clocks, also $(v, w') \models I(q_0)$. It follows that (q_0, w') is a state of the concrete semantics $[\![\mathcal{A}]\!]_v$ and $(q_0, w') \xrightarrow{d} (q_0, w)$. Since $w' \in [\![\mathsf{E}]\!]_v$, w' must be of the form $\lambda x.0$, so (q_0, w') is an initial state of the concrete semantics. This completes the proof of induction basis.

For the induction step, assume that we have a run in the symbolic semantics, ending with a transition $(q', C', D') \to (q, C, D)$. Using the fact that $(v, w) \in [\![C, D]\!]$, the definition of transitions in the symbolic semantics, Lemma 5 and Lemma 4, we know that there is a transition $q' \xrightarrow{a,g,r} q$ in $\mathcal{A}$, and there are C'', D'' such that $(v, w) \models I(q)$, $(v, w) \in [\![C'', D''[r]\uparrow]\!]$ and $(C', D') \overset{g}{\Leftarrow\!\!\shortmid} (C'', D'')$. By Lemma 9, we get that there exists a $d \geq 0$ and $w' \in [\![D''[r]]\!]_v$ such that $w' + d = w$. Since $(v, w) \models I(q)$ and invariants in a PTA only give upper bounds on clocks, also $(v, w') \models I(q)$. It follows that (q, w') is a state of the concrete semantics $[\![\mathcal{A}]\!]_v$ and $(q, w') \xrightarrow{d} (q, w)$. Using Lemma 8 we get that there exists a $w'' \in [\![D'']\!]_v$ such that $w' = w''[r]$. Using Lemma 5 and Lemma 4 again, it follows that $(v, w'') \models g$ and $(v, w'') \in [\![C', D']\!]$. Following Definition 10, we already observed that the location invariant holds for any reachable state in the symbolic semantics. In particular, $(v, w'') \models I(q')$. Hence, by definition of the concrete semantics, (q', w'') is a state of the concrete semantics and $(q', w'') \xrightarrow{a} (q, w')$ is a transition in the concrete semantics. By induction hypothesis, there is a path in the concrete semantics leading up to state (q', w''). Extension of this path with the transitions $(q', w'') \xrightarrow{a} (q, w')$ and $(q, w') \xrightarrow{d} (q, w)$ gives the required path in the concrete semantics.

For each path in the concrete semantics, we can find a path in the symbolic semantics such that the final state of the first path is semantically contained in the final state of the second path.

Proposition 2. *For each parameter valuation v and clock valuation w, if there is a run in the concrete semantics $[\![\mathcal{A}]\!]_v$ reaching a state (q, w), then this run can be simulated by a run in the symbolic semantics reaching a state (q, C, D) such that $(v, w) \in [\![C, D]\!]$.*

Proof. In any execution in the concrete semantics, we can always insert zero delay transitions at any point. Also, two consecutive delay transitions $(q, w) \xrightarrow{d} (q, w + d)$ and $(q, w + d) \xrightarrow{d'} (q, w + d + d')$ can always be combined to a single delay transition $(q, w) \xrightarrow{d+d'} (q, w + d + d')$. Therefore, without loss of generality, we only consider concrete executions that start with a delay transition, and in which there is a strict alternation of action

transitions and delay transitions. The proof is by induction on the number of action transitions.

As basis we consider a run $(q_0, w_0) \xrightarrow{d} (q_0, w_0 + d)$, where $w_0 = \lambda x.0$, consisting of a single time-passage transition. By definition of the concrete semanctics, $(v, w_0 + d) \models I(q_0)$. Using Lemma 9, we have that $(v, w_0 + d) \in [\![\top, \mathsf{E}\uparrow]\!]$ since $(v, w_0) \in [\![\top, \mathsf{E}]\!]$. From $(v, w_0 + d) \in [\![\top, \mathsf{E}\uparrow]\!]$ and $(v, w_0 + d) \models I(q_0)$, using Lemma 4 and Lemma 5 we get that there exists C, D such that $(\top, \mathsf{E}\uparrow) \overset{I(q_0)}{\Leftarrow\!\!\shortmid} (C, D)$ and $(v, w_0 + d) \in [\![C, D]\!]$. By definition, (C, D) is an initial state of the symbolic semantics. This completes the proof of the induction basis.

For the induction step, assume that the run in the concrete semantics of $[\![\mathcal{A}]\!]_v$ ends with transitions $(q'', w'') \xrightarrow{a} (q', w') \xrightarrow{d} (q, w)$. By induction hypothesis, there exists a run in the symbolic semantics ending with a state (q'', C'', D'') such that $(v, w'') \in [\![C'', D'']\!]$.

By definition of the concrete semantics, there is a transition $q'' \xrightarrow{g,a,r} q'$ in $\mathcal{A}$ such that $(v, w'') \models g$ and $w' = w''[r]$. Moreover, we have $q' = q$ and $w = w' + d$ and $(v, w) \models I(q)$. Using Lemma 4 and Lemma 5 gives that there exists C', D' such that $(C'', D'') \overset{g}{\Leftarrow\!\!\shortmid}_c (C', D')$ and $(v, w'') \in [\![C', D']\!]$. By Lemma 8, $w' \in [\![D'[r]]\!]_v$. Moreover, by Lemma 9, $w \in [\![D'[r]\uparrow]\!]_v$. Using $(v, w) \models I(q)$, Lemma 4 and Lemma 5, we infer that there exists C, D such that $(v, w) \in [\![C, D]\!]$ and $(C', D'[r]\uparrow) \overset{I(q)}{\Leftarrow\!\!\shortmid}_c (C, D)$. Finally, using the definition of the symbolic semantics, we infer the existence of a transition $(q'', C'', D'') \to (q, C, D)$.

Example 2. Figure 2 shows the symbolic state-space of the automaton in Fig. 1 represented by constrained PDBMs. In the initial state the invariant $x \leq p$ limits the value of x, and since both clocks have the same value also the value of y. When taking the transition from S0 to S1 we have to compare the parameters p and q. This leads to a split where in the one case no state is reachable since the region is empty, and in the other (when $q \leq p$) S1 can be reached. From then on no more splits occur and only one new state is reachable.

3.4 Evaluating Properties

We now define the relation $\overset{\phi}{\Leftarrow\!\!\shortmid}$ which relates a symbolic state and a state formula ϕ to a collection of symbolic states that satisfy ϕ.

In order to check whether a property holds, we break it down into the small basic formulas, namely checking locations and clock guards.

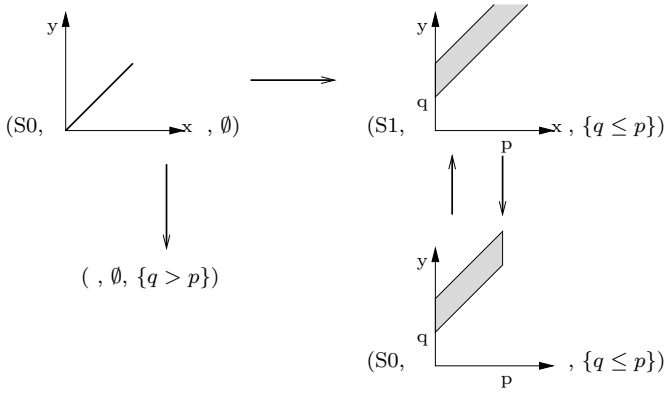

Fig. 3. The symbolic state space of the PTA in Fig. 1.

Checking that a clock guard holds relies on the definition given earlier, of adding that clock guard to the constrained PDBM. We rely on a special normal form of the state formula, in which all ¬ signs have been pushed down to the basic formulas.

Definition 11. *State formula ϕ is in* normal form *if all ¬ signs in ϕ appear only in front of a subformula that checks a location: $\neg q$.*

Since each simple guard with a ¬ sign in front can be rewritten to equivalent simple guard without, for each state formula there is an equivalent one in normal form.

In the following, let f be a simple guard, and ϕ be in normal form.

$$(Q_1)\frac{}{(q,C,D) \overset{q}{\Leftarrow\!\!\!\!\dashv} (q,C,D)} \qquad (Q_2)\frac{q \neq q'}{(q,C,D) \overset{\neg q'}{\Leftarrow\!\!\!\!\dashv} (q,C,D)}$$

$$(Q_3)\frac{(C,D) \overset{f}{\Leftarrow\!\!\!\!\dashv_c} (C',D')}{(q,C,D) \overset{f}{\Leftarrow\!\!\!\!\dashv} (q,C',D')}$$

$$(Q_4)\frac{(q,C,D) \overset{\phi_1}{\Leftarrow\!\!\!\!\dashv} (q,C',D') \ , \ (q,C',D') \overset{\phi_2}{\Leftarrow\!\!\!\!\dashv} (q,C'',D'')}{(q,C,D) \overset{\phi_1 \wedge \phi_2}{\Leftarrow\!\!\!\!\dashv} (q,C'',D'')}$$

$$(Q_5)\frac{(q,C,D) \overset{\phi_1}{\Leftarrow\!\!\!\!\dashv} (q,C',D')}{(q,C,D) \overset{\phi_1 \vee \phi_2}{\Leftarrow\!\!\!\!\dashv} (q,C',D')} \qquad (Q_6)\frac{(q,C,D) \overset{\phi_2}{\Leftarrow\!\!\!\!\dashv} (q,C',D')}{(q,C,D) \overset{\phi_1 \vee \phi_2}{\Leftarrow\!\!\!\!\dashv} (q,C',D')}$$

The following lemma gives the soundness and completeness of relation $\overset{\phi}{\leftrightarrows}$.

Lemma 10. *Let $[\![\phi, q]\!]$ denote the set $\{(v, w) \mid (q, w) \models \phi\}$. Then for all properties ϕ in normal form $[\![C, D]\!] \cap [\![q, \phi]\!] = \bigcup \{[\![C', D']\!] \mid (q, C, D) \overset{\phi}{\leftrightarrows} (q, C', D')\}$.*

Proof. $\subseteq$: We prove that, for all C, D, v, w and q, if $(v, w) \in [\![C, D]\!] \wedge (q, w) \models \phi$ then there are C', D' such that $(v, w) \in [\![C', D']\!] \wedge (q, C, D) \overset{\phi}{\leftrightarrows} (q, C', D')$. We use induction on $|\phi|$, where $|\phi|$ yields the depth of ϕ, as follows. For a location q and a simple guard f, we have $|q| = |\neg q| = |f| = 0$ and for composed properties we have $|\phi_1 \wedge \phi_2| = |\phi_1 \vee \phi_2| = 1 + \max(|\phi_1|, |\phi_2|)$.

- Base cases. Let $|\phi| = 0$ and let $(v, w) \in [\![C, D]\!]$ and $(q, w) \models \phi$.
  - Suppose $\phi = q'$. As $(q, w) \models q'$, clearly, $q = q'$. Since $(q, C, D) \overset{q}{\leftrightarrows} (q, C, D)$, we can take $C = C'$ and $D = D'$ and the result follows.
  - Suppose $\phi = \neg q'$. Similar to the previous case.
  - Suppose $\phi = f$ with f a simple guard. Then $(v, w) \in [\![C, D]\!]$ and $(q, w) \models f$. By Lemma 4 we have that there exist C'', D'' such that $(C, D) \overset{f}{\leftrightarrows} (C'', D'')$ and using that $(v, w) \in [\![C'', D'']\!]$ and Lemma 5 yields C', D' with $(C'', D'') \rightarrow_c (C', D')$ and $(v, w) \in [\![C', D']\!]$. Thus, we also have $(q, C, D) \overset{f}{\leftrightarrows} (q, C', D')$.
- Induction step. Let $|\phi| = n+1$ and let $(v, w) \in [\![C, D]\!]$ and $(q, w) \models \phi$.
  - Suppose $\phi = \phi_1 \wedge \phi_2$. Clearly, $(q, w) \models \phi_1$ and $(q, w) \models \phi_2$. By applying the induction hypothesis on ϕ_1, C and D, we derive that there exist C'', D'' such that $(q, C, D) \overset{\phi_1}{\leftrightarrows} (q, C'', D'')$ and $(v, w) \in [\![C'', D'']\!]$. Applying the induction hypothesis on ϕ_2, C'' and D'' yields C', D' such that $(q, C'', D'') \overset{\phi_2}{\leftrightarrows} (q, C', D')$ and $(v, w) \in [\![C', D']\!]$. Then also $(q, C, D) \overset{\phi_1 \wedge \phi_2}{\leftrightarrows} (q, C', D')$.
  - Suppose $\phi = \phi_1 \vee \phi_2$. Clearly, $(q, w) \models \phi_1$ or $(q, w) \models \phi_2$. Suppose $(q, w) \models \phi_1$. The induction hypothesis yields C', D' such that $(q, C, D) \overset{\phi_1}{\leftrightarrows} (q, C', D')$ and $(v, w) \in [\![C', D']\!]$. Then $(q, C, D) \overset{\phi_1 \vee \phi_2}{\leftrightarrows} (q, C', D')$. The case $(q, w) \models \phi_2$ is similar.

$\supseteq$: By induction on the structure of the derivation of $\overset{\phi}{\leftrightarrows}$, we establish that for all v, w, C, D, C', D', if $(q, C, D) \overset{\phi}{\leftrightarrows} (q, C', D')$ and $(v, w) \in [\![C', D']\!]$ then $(v, w) \in [\![C, D]\!]$ and $(q, w) \models \phi$.

- Base cases. The derivation consists of a single step r. Assume $(q, C, D) \overset{\phi}{\Leftarrow\!\!\shortmid} (q, C', D')$ and $(v, w) \in [\![C', D']\!]$.
  * $r = Q_1$. Then $\phi = q$, $C = C'$, $D = D'$. Then clearly, $(v, w) \in [\![C, D]\!]$ and $(q, w) \models q$.
  * $r = Q_2$. Similar to the previous case.
  * $r = Q_3$. Suppose $\phi = f$ with f a simple guard. Then $(q, C, D) \overset{\phi}{\Leftarrow\!\!\shortmid} (q, C', D')$ has been derived from $(C, D) \overset{f}{\Leftarrow\!\!\shortmid}_c (C', D')$. Then there exist C'', D'' such that $(C, D) \overset{f}{\Leftarrow\!\!\shortmid} (C'', D'')$ and $(C'', D'') \rightarrow_c (C', D')$. By Lemma 5 we have $(v, w) \in [\![C'', D'']\!]$. Then we have by Lemma 4 that $(v, w) \models f$ and $(v, w) \in [\![C, D]\!]$.
- Induction step. Assume $(q, C, D) \overset{\phi}{\Leftarrow\!\!\shortmid} (q, C', D')$ and $(v, w) \in [\![C', D']\!]$ and consider the last rule r used in the derivation of $(q, C, D) \overset{\phi}{\Leftarrow\!\!\shortmid} (q, C', D')$.
  * $r = Q_4$. Then $\phi = \phi_1 \wedge \phi_2$ and $(q, C, D) \overset{\phi_1}{\Leftarrow\!\!\shortmid} (q, C'', D'')$ and $(q, C'', D'') \overset{\phi_2}{\Leftarrow\!\!\shortmid} (q, C', D')$ for some C'', D''. Applying the induction hypothesis to the second statement yields that $(q, w) \models \phi_2$ and $(v, w) \in [\![C'', D'']\!]$. Then applying the induction hypothesis to the first statement yields $(q, w) \models \phi_1$ and $(v, w) \in [\![C, D]\!]$. Then also $(v, w) \models \phi_1 \wedge \phi_2$.
  * $r = Q_5$. Then $\phi = \phi_1 \vee \phi_2$. Then $(q, C, D) \overset{\phi_1}{\Leftarrow\!\!\shortmid} (q, C', D')$. By the induction hypothesis we have that $(q, w) \models \phi_1$ and $(v, w) \in [\![C, D]\!]$.
  * $r = Q_6$. Similarly to the previous case.

3.5 Algorithm

We are now in a position to present our model checking algorithm for parametric timed automata. The algorithm displayed in Fig. 4 describes how our tool explores the symbolic state space and searches for constraints on the parameters for which a reachability formula $\exists\Diamond\phi$ holds in a PTA $\mathcal{A}$. The result returned by the algorithm is a set of symbolic states, all of which satisfy ϕ, for any valuation of the parameters and clocks in the state. For invariance properties $\forall\Box\phi$, the tool performs the algorithm on $\neg\phi$, and the result is then a set of symbolic states, none of which satisfies ϕ. The answer to the model checking problem, stated in Section 2.2, is obtained by taking the union of the constraint sets from all symbolic states in the result of the algorithm; in the case of an invariance property we take the complement of this set.

algorithm Reachable($\mathcal{A}$, ϕ)

RESULT := $\emptyset$,PASSED := $\emptyset$,WAITING := $\{(q_0, C, D) \mid (\top, \mathsf{E}\uparrow) \overset{I(q_0)}{\Leftarrow}_c (C, D)\}$
while WAITING $\neq \emptyset$ **do**
select (q, C, D) from WAITING
RESULT := RESULT $\cup$ $\{(q', C', D') \mid (q, C, D) \overset{\phi}{\Leftarrow} (q', C', D')\}$
FALSE := $\{(q', C', D') \mid (q, C, D) \overset{\neg\phi}{\Leftarrow} (q', C', D')\}$
for each (q', C', D') in FALSE **do**
if for all (q'', C'', D'') in PASSED: $(q', C', D') \not\subseteq (q'', C'', D'')$ **then**
add (q', C', D') to PASSED
for each (q'', C'', D'') such that $(q', C', D') \rightarrow (q'', C'', D'')$ **do**
WAITING := WAITING $\cup$ $\{(q'', C'', D'')\}$
return RESULT

Fig. 4. The parametric model checking algorithm

A difference between the above algorithm and the standard timed model checking algorithm is that we continue the exploration until either no more new states are found or all paths end in a state satisfying the property. This is because we want to find all the possible constraints on the parameters for which the property holds. Also, the operations on non-parametric DBMs only change the DBM they are applied to, whereas in our case, we may end up with a set of new PDBMs and not just one.

Some standard operations on symbolic states that help in exploring as little as possible, have also been implemented in our tool for parametric symbolic states. Before starting the state space exploration, our implementation determines the *maximal constant* for each clock. This is the maximal value to which the clock is compared in any guard or invariant in the PTA. When the clock value grows beyond this value, we can ignore its real value. This enables us to identify many more symbolic states, and helps termination[2].

4 Reducing the Complexity

This section introduces the class of lower bound/upper bound automata and describes several (rather intuitive) observations that simplify the model checking of PTAs in this class. Our results allow us to eliminate parameters in certain cases. Since the complexity of parametric model checking grows very fast in the number of parameters, this is a relevant issue. Secondly, our observations yield a decidability result for lower

[2] For purely timed model checking this *guarantees* termination.

bound/upper bound automata whereas the corresponding problem for general PTAs is undecidable.

Informally, a positive occurrence of a parameter in a guard or an invariant of a PTA enforces (or contributes to) an upper bound on a clock difference, for instance p in $x - y < 2p$. A negative occurrence of a parameter contributes to a lower bound on a clock difference, for instance q and q' in $y - x > q + 2q'$ ($\equiv x - y < -q - 2q'$) and in $x - y < 2p - q - 2q'$. Hence, a PTA containing the guards $x - y \leq p - q$ and $z < q - p$ is not an L/U automaton.

Definition 12. *A parameter $p_i \in P$ is said to* occur *in the linear expression $e = t_0 + t_1 \cdot p_1 + \cdots t_n \cdot p_n$ if $t_i \neq 0$; p_i* occurs positively *in e if $t_i > 0$ and p_i* occurs negatively *in e if $t_i < 0$. A* lower bound parameter *of a PTA $\mathcal{A}$ is a parameter that only occurs negatively in the expressions of $\mathcal{A}$ and an* upper bound parameter *of $\mathcal{A}$ a parameter that only occurs positively in $\mathcal{A}$. We call $\mathcal{A}$ a* lower bound/upper bound (L/U) automaton *if every parameter occuring in $\mathcal{A}$ is either a lower bound parameter or an upper bound parameter of $\mathcal{A}$, but not both.*

Example 3. The automaton in Fig. 5 is an L/U automaton where min is a lower bound parameter and max is an upper bound parameter. Also the model of Fischer protocol in Fig. 2 is an L/U automaton. Here min_rw and min_delay are lower bound parameters and max_rw and max_delay are the upper bound parameters.

From now on, we work with a fixed set $L = \{l_1, \ldots l_K\}$ of lower bound parameters and a fixed set $U = \{u_1, \ldots u_M\}$ of upper bound parameters with $L \cap U = \emptyset$ and $L \cup U = P$.

We consider, apart from parameter valuations, also *extended parameter valuations*. Intuitively, an extended parameter valuation is a parameter valuation with values in $\mathsf{R}^{\geq 0} \cup \{\infty\}$, rather than in $\mathsf{R}^{\geq 0}$. Extended parameter valuations are useful in certain cases to solve the verification problem (over non-extended valuations) stated in Section 2.3. Working with extended parameter valuations may cause the evaluation of an expression to be undefined. For example, the expression $e[v]$ is not defined for $e = p - q$ and $v(p) = v(q) = \infty$. We require that an extended parameter valuation of an L/U automaton does not assign the value ∞ both to a lower bound parameter and to an upper bound parameter. Then the expression $e[v]$ is defined for every extended valuation of an L/U automaton.

Therefore, we can easily extend notions $e[v]$, $(v, w) \models e$ and $\mathcal{A}[v]$ (defined in Section 2) to extended valuations, by using the conventions

that $0 \cdot \infty = 0$, that $x - y \prec \infty$ evaluates to true and $x - y \prec -\infty$ to false. In particular, we have $[\![\mathcal{A}]\!]_v = \mathcal{A}[v]$ for extended valuations v and L/U automata $\mathcal{A}$. Moreover, we extend the orders $\sim$ to $\mathsf{R} \cup \{\infty\}$ in the usual way and to extended valuations via point wise extension (i.e. $v \sim v'$ iff $v(p) \sim v'(p)$ for all $p \in P$). We denote an extended valuation of an L/U automaton by a pair (λ, μ), which equals the function λ on the lower bound parameters and μ on the upper bound parameters. We write 0 and ∞ for the functions assigning respectively 0 and ∞ to each parameter.

The following proposition is based on the fact that weakening the guards in $\mathcal{A}$ (i.e. decreasing the lower bounds and increasing the upper bounds) yields an automaton whose reachable states include those of $\mathcal{A}$. Dually, strengthening the guards in $\mathcal{A}$ (i.e. increasing the lower bounds and decreasing the upper bounds) yields an automaton whose reachable states are a subset of those of $\mathcal{A}$. We claim that this proposition, formulated for L/U automata, can be generalized to lower bound and upper bound parameters present in general PTAs. It is however crucial that (by definition) state formulae do not contain parameters. The usefulness of this property (and of several other properties in this section) lies in the fact that it allows to conclude the satisfaction of a property for infinitely many parameter valuations from the satisfaction of that property for one valuation.

Proposition 3. *Let $\mathcal{A}$ be an L/U automaton and ϕ a state formula. Then*

1. $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \models \exists\Diamond\phi \iff \forall \lambda' \leq \lambda, \mu \leq \mu' : [\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \exists\Diamond\phi$.
2. $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \models \forall\Box\phi \iff \forall \lambda \leq \lambda', \mu' \leq \mu : [\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \forall\Box\phi$.

Proof. (sketch) Both parts of the proposition can be proven by induction on the length of runs in the L/U automata. The crucial observation is that for parameter valuations $\lambda' \leq \lambda$ and $\mu \leq \mu'$ and linear expression e we have that $e[\lambda', uval] \leq e[lval, uval]$ and $e[\lambda, uval] \leq e[\lambda, uvalpr]$. Therefore whenever $((\lambda, \mu), w) \models g$ then $((\lambda', \mu'), w) \models g$. □

The following example illustrates how Proposition 3 can be used to eliminate parameters from L/U automata.

Example 4. The automaton in Fig. 5 is an L/U automaton. Its location S_1 is reachable irrespective of the parameter values. By setting the parameter *min* to ∞ and *max* to 0, one checks with a non-parametric model checker that $\mathcal{A}[(\infty, 0)] \models \exists\Diamond S_1$. Then Proposition 3 (together with $[\![\mathcal{A}]\!]_v = \mathcal{A}[v]$) yields that S_1 is reachable in $[\![\mathcal{A}]\!]_{(\lambda,\mu)}$ for all extended parameter valuations $0 \leq \lambda, \mu \leq \infty$.

Clearly, $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \models \exists\Diamond S_2$ iff $\lambda(min) \leq \mu(max) \wedge \lambda(min) < \infty$. We will see in this running example how we can verify this property completely by non-parametric model checking. Henceforth, we construct the automaton $\mathcal{A}'$ from $\mathcal{A}$ by substituting the parameter *max* by the parameter *min* yielding an (non L/U) automaton with one parameter, *min*. If we show that $[\![\mathcal{A}']\!]_v \models \exists\Diamond S_2$ for all valuations v, this essentially means that $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \models \exists\Diamond S_2$ for all λ, μ such that $\mu(max) = \lambda(min) < \infty$ and then Proposition 3 implies that $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \models \exists\Diamond S_2$ for all λ, μ with $\lambda(min) \leq \mu(max)$ and $\lambda(min) < \infty$.

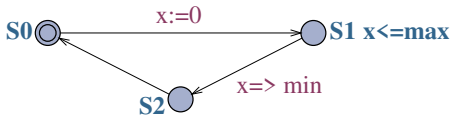

Fig. 5. Reducing parametric to non-parametric model checking

The question whether there exists a (non-extended) parameter valuation such that a given (final) location q is reachable, is known as the *emptiness problem* for PTAs. In [3], it is shown that the emptiness problem is undecidable for PTAs with three clocks or more. The following proposition implies that we can solve the emptiness problem for an L/U automaton $\mathcal{A}$ by only considering $\mathcal{A}[(0,\infty)]$, which is a non-parametric timed automaton. Since reachability is decidable for timed automata ([2]), the emptiness problem is decidable for L/U automata. Then it follows that the dual problem is also decidable for L/U automata. This is the *universality problem* for invariance properties, asking whether an invariance property holds for all parameter valuations.

Proposition 4. *Let $\mathcal{A}$ be an L/U automaton. Then $\mathcal{A}[(0,\infty)] \models \exists\Diamond q$ if and only if there exist a (non-extended) parameter valuation (λ, μ) such that $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \models \exists\Diamond q$.*

Proof. The "if"–part is an immediate consequence of Proposition 3 and the fact that $\mathcal{A}[(0,\infty)] = [\![\mathcal{A}]\!]_{(0,\infty)}$. For the "only if"–part, assume that α is a run of $\mathcal{A}[(0,\infty)]$ that reaches q. Let T' be the smallest constant occurring $\mathcal{A}$ and T be the maximum clock value occurring in α. (More precisely, if $\alpha = s_0 a_1 s_1 a_2 \ldots a_N s_N$ and $s_i = (q_i, w_i)$, then $T = \max_{i \leq N, x \in C} w_i(x)$; T' compensates for negative constants t_0.) Now, take $\lambda(l_j) = 0$ and

$\mu(u_j) = T + |T'| + 1$. Then for every guard or invariant g occurring in $\mathcal{A}$ we have that $((0,\infty), w_i) \models g \implies ((\lambda,\mu), w_i) \models g$. Hence, α is a run of $[\![\mathcal{A}]\!]_{(\lambda,\mu)}$, so $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \models \exists\Diamond q$. □

Corollary 1. *The emptiness problem is decidable for L/U automata.*

Definition 13. *A PTA $\mathcal{A}$ is* fully parametric *if clocks are only reset to* 0 *and every linear expression in $\mathcal{A}$ of the form $t_1 \cdot p_1 + \cdots + t_n \cdot p_n$, where $t_i \in \mathsf{Z}$.*

The following proposition is basically the observation in [2], that multiplication of each constant in a timed automaton and in a system property with the same positive factor preserves satisfaction.

Proposition 5. *Let $\mathcal{A}$ be fully parametric PTA. Then for all parameter valuations v and all system properties ψ*

$$[\![\mathcal{A}]\!]_v \models \psi \iff \forall t \in \mathsf{R}^{>0} : [\![\mathcal{A}]\!]_{t\,\cdot\, v} \models t \cdot \psi,$$

where $t \cdot v$ denotes the valuation $p \mapsto t \cdot v(p)$ and $t \cdot \psi$ the formula obtained from ψ by multiplying each number in ψ by t.

Proof. It is easy to see that for all $t \in \mathsf{R}^{>0}$, $\alpha = s_0 a_1 s_1 a_2 \ldots a_N s_N$ with $s_i = (q_i, w_i)$ is a run of $[\![\mathcal{A}]\!]_v$ if and only if $s'_0 a_1 s'_1 \ldots a_N s'_N$ is a run of $[\![\mathcal{A}]\!]_{t\,\cdot\, v}$, where $s'_i = (q_i, t \cdot w_i)$ and $t \cdot w_i$ denotes $x \mapsto t \cdot w_i(x)$.

Then for fully parametric PTAs with one parameter and system properties ψ without constants (except for 0), we have $[\![\mathcal{A}]\!]_v \models \psi$ for all valuations v of P if and only if both $\mathcal{A}[0] \models \psi$ and $\mathcal{A}[1] \models \psi$. The fact that the 0-case has to be treated separatly is illustrated by the (fully parametric) automaton with a single transition equipped with the guard $x < p$. The target location of the transition is not reachable for $p = 0$.

Corollary 2. *For fully parametric PTAs with one parameter and properties ψ without constants (except 0), it is decidable whether $\forall v \in [\![C]\!] : [\![\mathcal{A}]\!]_v \models \psi$.*

Example 5. The PTA $\mathcal{A}'$ mentioned in Example 4 is a fully parametric timed automaton and the property $\exists\Diamond S_2$ is without constants. We establish that $\mathcal{A}'[0] \models \exists\Diamond S_2$ and $\mathcal{A}'[1] \models \exists\Diamond S_2$. Then Proposition 5 implies that $\mathcal{A}'[v] \models \exists\Diamond S_2$ for all v. As shown in Example 4, this implies that $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \models \exists\Diamond S_2$ for all λ, μ with $\lambda(min) = \mu(max) < \infty$.

In the running example, we would like to use the same methods as above to verify that $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \not\models \exists\Diamond S_2$ if $\lambda(min) > \mu(max)$. We can in this case not fill in for $min = max$, since the bound in the constraint is a strict one. The following definition and results allows us to move the strictness of a constraint into the PTA.

Definition 14. *Let $P' \subseteq P$ be a set of parameters. Define $\mathcal{A}^{\leq}_{P'}$ as the automaton obtained by replacing every inequality $x-y \leq e$ in $\mathcal{A}$ by a strict inequality $x-y < e$, provided that e contains at least one parameter from P'. Similarly, define $\mathcal{A}^{<}_{P'}$ as the automaton from $\mathcal{A}$ obtained by replacing every inequality $x-y < e$ by a non–strict inequality $x-y \leq e$, provided that e contains at least one parameter from P'. For $\prec = <, \leq$, write $\mathcal{A}^{\prec}$ for $\mathcal{A}^{\prec}_{P}$. Moreover, define $v \prec_{P'} v'$ by $v(p) \prec v'(p)$ if $p \in P'$ and $v(p) = v'(p)$ otherwise.*

Proposition 6. *Let $\mathcal{A}$ be an L/U automaton. Then*

1. $[\![\mathcal{A}^{\leq}]\!]_{(\lambda,\mu)} \models \exists\Diamond\phi \implies \forall \lambda' < \lambda, \mu < \mu' : [\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \exists\Diamond\phi$.
2. $[\![\mathcal{A}^{<}]\!]_{(\lambda,\mu)} \models \forall\Box\phi \iff \forall \lambda < \lambda', \mu' < \mu : [\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \forall\Box\phi$.

Proof. **1,** $\implies$ Let e be a linear expression occuring in $\mathcal{A}$. Then we can write $e = t_0 + e_1 + e_2$, where $t_0 \in \mathsf{Z}$, e_1 is an expression over the upperbound parameters and e_2 an expression over the lower bound parameters. Then we have

$$\begin{aligned}
&\mu \leq \mu' \implies e_1[\mu] \leq e_1[\mu'],\\
&\lambda' \leq \lambda \implies e_2[\lambda'] \leq e_2[\lambda],\\
&\lambda' \leq \lambda, \mu \leq \mu' \implies e[(\lambda,\mu)] \leq\ e[(\lambda',\mu')].
\end{aligned}$$

If there is at least one parameter occuring in e_1 or e_2 respectively then respectively

$$\begin{aligned}
\mu < \mu' &\implies e_1[\mu] < e_1[\mu']\\
\lambda' < \lambda &\implies e_2[\lambda] < e_2[\lambda'].
\end{aligned}$$

Thus if there is at least one parameter occuring in e, then

$$\lambda' < \lambda, \mu < \lambda \implies e[(\lambda,\mu)] < e[(\lambda',\mu')].$$

Now, let (λ,μ) be an extended valuation. Let $g \equiv x-y \prec e$ be a simple guard occuring in $\mathcal{A}^{\leq}$ and let $g' \equiv x-y \prec' e$ be the corresponding guard in $\mathcal{A}$. Assume that $(w,(\lambda,\mu)) \models g$, i.e. $w(x)-w(y) \prec e[(\lambda,\mu)]$. We show that $(w,(\lambda,\mu)) \models g'$. We distinguish two cases.

case 1: There exists a parameter occuring in e. Then $w(x) - w(y) \prec e[(\lambda, \mu)] < e[(\lambda', \mu')]$. Then certainly $(w, (\lambda, \mu)) \models g' \equiv x - y \prec' e$.

case 2: The expression e does not contain any parameter. Then $g' \equiv g$ and hence $(w, (\lambda, \mu)) \models g'$.

Now it easily follows that every run of $[\![\mathcal{A}^{\leq}]\!]_{(\lambda,\mu)}$ is also a a run of $[\![\mathcal{A}]\!]_{(\lambda',\mu')}$. Thus, if a state satisfying ψ is reachable in $[\![\mathcal{A}^{\leq}]\!]_{(\lambda,\mu)}$ then it is also reachable in $[\![\mathcal{A}]\!]_{(\lambda',\mu')}$.

2, $\Longrightarrow$: This follows from 1. Assume that $[\![\mathcal{A}^{<}]\!]_{(\lambda,\mu)} \models \forall\Box\phi$ and let λ', μ' be such that $\lambda < \lambda'$, $\mu' < \mu$. Since $[\![\mathcal{A}^{<}]\!]_{(\lambda,\mu)} \not\models \exists\Diamond\neg\phi$, we have

$$\neg\forall\lambda'' < \lambda', \mu' < \mu'' : [\![\mathcal{A}^{<}]\!]_{(\lambda'',\mu'')} \models \exists\Diamond\neg\phi.$$

Then contraposition of statement (1) of this proposition together with $(\mathcal{A}^{<})^{\leq} = \mathcal{A}^{\leq}$ yields $[\![\mathcal{A}^{\leq}]\!]_{(\lambda',\mu')} \not\models \exists\Diamond\neg\phi$. As $\mathcal{A}$ imposes stronger bounds than $\mathcal{A}^{\leq}$, also $[\![\mathcal{A}]\!]_{(\lambda',\mu')} \not\models \exists\Diamond\neg\phi$, i.e. $[\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \forall\Box\phi$.

2, $\Longleftarrow$: Let (λ, μ) be an extended valuation and assume that $[\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \forall\Box\phi$ for all $\lambda' > \lambda$, $\mu' < \mu$. Assume that α is a run of $[\![\mathcal{A}^{<}]\!]_{(\lambda,\mu)}$. We construct $\lambda' > \lambda$ and $\mu' < \mu$ such that α is also a run of $[\![\mathcal{A}]\!]_{(\lambda',\mu')}$. (Then we are done: since $[\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \forall\Box\phi$, the last state of α satisfies ϕ. Hence every reachable state of $[\![\mathcal{A}]\!]_{(\lambda,\mu)}$ satisfies ϕ, i.e. $[\![\mathcal{A}]\!]_{(\lambda,\mu)} \models \forall\Box\phi$.)

We use the following notation. We write $v = (\lambda, \mu)$ and $v' = (\lambda', \mu')$. For a run α, we write $\alpha = s_0 a_1 s_1 a_2 \dots a_N s_N$ with $s_i = (q_i, w_i)$, $I(q_i) = \wedge_j^{J'} I_{ij}$, $I_{ij} = x_{ij} \prec_{ij} E_{ij}$. As α is a run, there exists a transition $q_{i-1} \xrightarrow{g_i, a_i, r_i} q_i$ for each i, $1 \leq i \leq N$. We write the guard on this transition by $g_i = \wedge_j^{J} g_{ij}$, $g_{ij} = x_{ij} - y_{ij} \prec_{ij} e_{ij}$. Finally, if g is a guard or invariant in $\mathcal{A}$, then we denote the corresponding guard or invariant in $\mathcal{A}^{<}$ by $g^{<}$, i.e. the guard that is obtained as in Definition 14.

If neither the guards g_{ij} nor the invariants I_{ij} contains a parameter, then we can take v' arbitrarily and we have that α is a run of $[\![\mathcal{A}]\!]_{v'}$. Therefore, assume that at least one of the guards g_{ij} or invariants I_{ij} contains a parameter. Then, by definition of $\mathcal{A}^{<}$, this guard or invariant contains a strict bound. In this case, we construct $\lambda' > \lambda$ and $\mu' < \mu$ such that $w_i(x - y) < e[(\lambda', \mu')] < e[(\lambda, \mu)]$ if $g = x - y < e$ is an invariant I_{ij} or guard g_{ij} as above. Informally, we use the minimum "distance" $e[(\lambda, \mu)] - w_i(x - y)$ occurring in α to slightly increase the lower bounds and slightly decrease the upper bounds yielding $\lambda < \lambda'$ and $\mu < \mu'$.

Let

$$\begin{aligned} T_0 &= \min_{i\leq N, j\leq J'} \{E_{ij}[v] - w_i(x_{ij}) \quad | \quad \prec_{ij} = <\}, \\ T_1 &= \min_{i\leq N, j\leq <J} \{e_{ij}[v] - w_i(x_{ij} - y_{ij}) \quad | \quad \prec_{ij} = <\}, \\ 0 &< T < \min\{T_0, T_1\}, \end{aligned}$$

with the convention that $\min \emptyset = \infty$. At least one of the inequalities $\prec_i j$ is strict, since at least one of the guards contains a parameter. Hence $T_0 < \infty$ or $T_1 < \infty$. Since $(v, w_i) \models I_{ij} \wedge g_{ij}$, we have we have that $T_0 \geq 0$ and $T_1 \geq 0$. Hence $\infty > \min\{T_0, T_1\} > 0$ and the requested T exists. It crucial property is that if $g_{ij} \equiv x_{ij} - y_{ij} < e_{ij}$ or $g_{ij} \equiv x_{ij} - y_{ij} < E_{ij}$ we have respectively

$$\begin{aligned} T &< e_{ij}[v] - w_i(x_{ij} - y_{ij}) \\ T &< E_{ij}[v] - w_i(x_{ij} - y_{ij}). \end{aligned}$$

Now, let T' be the sum of the constants appearing in the guards and invariants that appear in the run α i.e.

$$T' = \sum_{i\leq N, j\leq J'} \mathrm{sum_of_const(E_{ij})} + \sum_{\mathrm{i\leq n, j\leq J}} \mathrm{sum_of_const(e_{ij})},$$

where $\mathrm{sum_of_const(t_0 + t_1 \cdot p_1 + \cdots + t_n \cdot p_n)} = |\mathrm{t_1}| + \cdots + |\mathrm{t_n}|$. Since at least one of the guards or invariants contains a parameter, we have $T' > 0$.
Now, take $v' = (\lambda + \frac{T}{T'}, \mu - \frac{T}{T'})$ and consider $g_{ij} \equiv x_{ij} - y_{ij} \prec_{ij} e_{ij}$. We claim that $(v', w_i) \models g_{ij}$.

case 1: The expression g_{ij} does not contain any parameter. Then $g_{ij} = g_{ij}^{<}$ and $e_{ij}[v] = e_{ij}[v']$. Since $(w_i, v) \models g_{ij}$, also $(w_i, (v')) \models g_{ij}^{<}$.

case 2: There exists a parameter occuring in e. We can write $e = t_0 + t_1 \cdot u_1 + \cdots + t_M \cdot u_M - t'_1 \cdot l_1 - \cdots - t'_K \cdot l_K$, with $t_i \geq 0$, $t'_i \geq 0$ for $i > 0$. Then

$$\begin{aligned} e_{ij}[v'] &= t_0 + \sum_{k=1}^{M} t_k \cdot (\mu'_k - \frac{T}{T'}) - \sum_{k=1}^{K} t_k \cdot (\lambda'_k + \frac{T}{T'}) \\ &= (t_0 + \sum_{k=1}^{M} t_k \cdot \mu'_k - \sum_{k=1}^{K} t_k \cdot \lambda'_k) - \frac{T}{T'} \cdot (\sum_{k=1}^{M} t_k + \sum_{k=1}^{K} t'_k) \\ &\geq e_{ij}[v] - T \\ &> e_{ij}[v] - (e_{ij}[v] - w_i(x_{ij} - y_{ij})) \\ &= w_i(x_{ij} - y_{ij}). \end{aligned}$$

Therefore $(w_i, v') \models x_{ij} - y_{ij} < g_{ij}^{\leq}$ and then also $(w_i, v') \models x_{ij} - y_{ij} \prec_{ij} g_{ij}^{\leq}$.

Combining the cases (1) and (2) yields that for all i, j, $(w_i, v') \models x_{ij} - y_{ij} \prec_{ij} g_{ij}^{\leq}$. Similarly, one proves that $(w_i, v') \models x_{ij} - y_{ij} \prec_{ij} I_{ij}$. Therefore, α is a fun of $[\![\mathcal{A}]\!]_{(\lambda',\mu')}$.

□

The previous result concerns the automaton that is obtained when all the strict inequalities in the automaton are changed into nonstrict ones, (or the other way around). Sometimes, we want to 'toggle' only a some of the inequalities. Then the following result can be applied.

Corollary 3. *Let $\mathcal{A}$ be an L/U automaton and $P' \subseteq P$.*

1. $[\![\mathcal{A}^{\leq}_{P'}]\!]_{(\lambda,\mu)} \models \exists\Diamond\phi \implies \forall\lambda' <_{P'} \lambda, \mu <_{P'} \mu' : [\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \exists\Diamond\phi.$
2. $[\![\mathcal{A}^{<}_{P'}]\!]_{(\lambda,\mu)} \models \forall\Box\phi \iff \forall\lambda <_{P'} \lambda', \mu' <_{P'} \mu : [\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \forall\Box\phi.$

Proof. Let (λ, μ) be an extended valuation. Let $\mathcal{A}_0$ be the automaton obtained from $\mathcal{A}$ by substituting p by $(\lambda, \mu)(p)$ for every $p \notin P'$. Then $\mathcal{A}^{<}_{P'} = \mathcal{A}_0{}^{<}$ and $\mathcal{A}^{\leq}_{P'} = \mathcal{A}_0{}^{\leq}$. Now the result follows by applying Proposition 6 to $\mathcal{A}_0$. □

The following example shows that the converse of Proposition 6, item 1 does not hold.

Example 6. Consider the automaton $\mathcal{A}$ in Fig. 6. Recall that the clocks x and y are initially 0. Then $\mathcal{A} = \mathcal{A}^{\leq}$ and the location q is reachable if $max > 0$ but not if $max = 0$. Thus $\forall\lambda' < 0, 0 < \mu' : [\![\mathcal{A}]\!]_{(\lambda',\mu')} \models \exists\Diamond\phi$, but not $[\![\mathcal{A}^{\leq}]\!]_{(0,0)} \models \exists\Diamond\phi$.

We believe the class of L/U automata can be very useful in practice. Several examples known from the literature fall into this class, or can be modelled slightly differently to achieve this. We mention the IEEE Root Contention protocol [12], Fischer's mutual exclusion protocol [13], the (toy) rail road crossing example from [3], the Bounded Retransmission protocol (when considering a fixed value for the integer variables) and the Biphase Mark protocol (with minor adaptations). Moreover, the MMT models from [16] can be encoded straightforwardly into L/U automata.

We expect that quite a few other distributed systems and protocols can be modelled with L/U automata, since it is quite natural to have the duration of an event (such as the communication delay in a channel, the computation time needed to produce a result, the time required to

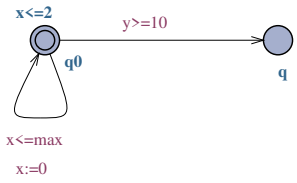

Fig. 6. The converse of Proposition 6 (1) does not hold.

open the gate in a rail road crossing) lying between a lower bound and an upper bound and these bounds are often the parameters of the system.

Section 4.1 and Section 5 show that the techniques discussed in this section to eliminate parameters in L/U models reduce the verification effort significantly and possibly leads to a completely non-parametric model.

4.1 Verification of Fischer's Mutual Exclusion Protocol

In this section, we apply the results from the previous section to verify the Fischer protocol with 2 processes. We can establish the sufficiency of the protocol constraints by non-parametric model checking and the necessity of the constraints by eliminating three of the four parameters.

Consider the Fischer protocol from Section 2.4 again. In this section, we consider a system $\mathcal{A}$ consisting of two parallel processes P_1 and P_2. It is clear that $\mathcal{A}$ is a fully parametric L/U automaton: min_rw and min_delay are lower bound parameters and max_rw and max_delay upper bound parameters.

The mutual exclusion property is expressed by the formula $\Phi_{ME} \equiv \forall\Box\neg(P_1.cs \wedge P_2.cs)$. Recall that assuming the basic constraints $B_{ME} \equiv 0 \leq min_rw < max_rw \ \wedge \ 0 \leq min_delay < max_delay$, mutual exclusion is guaranteed if and only if $C_{ME} \equiv max_rw \leq min_delay$. Thus we prove that for all valuations v: $v \models B_{ME} \implies (\llbracket\mathcal{A}\rrbracket_v \models \Phi_{ME} \iff v \models C_{ME})$.

Sufficiency of the Constraints We show that the constraints assure mutual exclusion, that is

$$\text{if } v \models C_{ME} \wedge B_{ME}, \text{ then } \mathcal{A}[v] \models \Phi_{ME}.$$

We perform the substitution

$$min_rw \mapsto 0, max_delay \mapsto \infty, min_delay \mapsto max_rw$$

to obtain a fully parametric automaton $\mathcal{A}'$ with one parameter, max_rw. We have established by non-parametric model checking that $\mathcal{A}'[0] \models \Phi_{ME}$ and $\mathcal{A}'[1] \models \Phi_{ME}$. Now Proposition 5 yields that $[\![\mathcal{A}']\!]_v \models \Phi_{ME}$ for all valuations v (where only the value of max_delay matters). This means that $[\![\mathcal{A}]\!]_v \models \Phi_{ME}$ if $v(min_rw) = 0$, $v(max_rw) = v(min_delay)$ and $v(max_delay) = \infty$. Then Proposition 3 yields that the mutual exclusion property, which is an invariance property, also holds if the lower bound parameters min_rw and min_delay are increased and if the upper bound parameter max_rw is decreased. More precisely, Proposition 3 implies that $[\![\mathcal{A}]\!]_v \models \Phi_{ME}$ for all v with $0 \leq v(min_rw)$, $v(max_rw) \leq v(min_delay)$ and $v(max_delay) \leq \infty$. Then, in particular, if $v \models C_{ME} \wedge B_{ME}$, then $[\![\mathcal{A}]\!]_v \models \Phi_{ME}$.

Necessity of the Constraints: We show that if

$$v \models B_{ME} \wedge \neg C_{ME} \implies \mathcal{A}[v] \models \neg\Phi_{ME},$$

i.e. that if $v \models min_rw < max_rw \wedge min_delay < max_delay \wedge min_delay < max_rw$, then $\mathcal{A}[v] \models \neg\Phi_{ME} \equiv \exists\Diamond(P_1.cs \wedge P_2.cs)$. We construct the automaton $\mathcal{A}^{\leq}$ and proceed in two steps.

Step 1 Let v_0 be the valuation $v_0(min_delay) = v_0(max_delay) = 0$ and $v_0(min_rw) = v_0(max_delay) = 1$. By non-parametric model checking we have established that

$$\mathcal{A}^{\leq}[0] \models \neg\Phi_{ME} \tag{7}$$

$$\mathcal{A}^{\leq}[v_0] \models \neg\Phi_{ME}. \tag{8}$$

We show that it follows that for all v

$$v \models 0 = min_delay = max_delay \leq min_rw = max_rw \implies \mathcal{A}^{\leq}[v] \models \neg\Phi_{ME}. \tag{9}$$

Assume $v \models 0 = min_delay = max_delay \leq min_rw = max_rw$. Consider $t = v(min_rw)$. If $v(min_rw) = 0$, then (7) shows that $[\![\mathcal{A}^{\leq}]\!]_v \models \neg\Phi_{ME}$. Therefore, assume $v(min_rw) > 0$ and consider $\frac{v}{t} \equiv \lambda x.\frac{v(x)}{t}$. It is not difficult to see that

$$\frac{v}{t} \models 0 = min_delay = max_delay \leq min_rw = max_rw = 1.$$

Therefore, (8) yields $[\![\mathcal{A}^{\leq}]\!]_{\frac{v}{t}} \models \neg\Phi_{ME}$. Since $\mathcal{A}^{\leq}$ is a fully parametric PTA, Proposition 5 yields that $[\![\mathcal{A}^{\leq}]\!]_v \models \neg\Phi_{ME}$.

Step 2 Let $\mathcal{A}'$ be the automaton that is constructed from $\mathcal{A}^{\leq}$ by performing the following substitution $min_delay \mapsto 1$, $max_delay \mapsto 1$, $min_rw \mapsto max_rw$. By parametric model checking we have established

$$v \models 1 \leq max_rw \implies [\![\mathcal{A}']\!]_v \models \neg\Phi_{ME}. \tag{10}$$

This means that if

$$v \models min_delay = max_delay = 1 \leq min_rw = max_rw \implies [\![\mathcal{A}^{\leq}]\!]_v \models \neg\Phi_{ME}.$$

By a argument similar to the one we used to prove (9), (where now the case $v(min_delay) = 0$ is covered by statement (9) in Step 1.), we can use Proposition 5 to show that

$$v \models min_delay = max_delay \leq min_rw = max_rw \implies [\![\mathcal{A}^{\leq}]\!]_v \models \neg\Phi_{ME}.$$

Now, Proposition 3 yields that $\neg\Phi_{ME}$ – which is a reachability property – also holds if the values for the lower bounds are decreased and the values for the upper bounds are increased. Note that we may increase max_delay as much as we want; $v(max_delay)$ may be larger than $v(min_rw)$. Thus we have

$$v \models min_rw \leq max_rw \ \wedge\ min_delay \leq max_delay \ \wedge\ min_delay \leq max_rw \implies [\![\mathcal{A}^{\leq}]\!]_v \models \neg\Phi_{ME}$$

and then Proposition 6 yields that

$$v \models min_rw < max_rw \ \wedge\ min_delay < max_delay \ \wedge\ min_delay < max_rw \implies [\![\mathcal{A}]\!]_v \models \neg\Phi_{ME}.$$

We have checked the result formulated in statement 10 with our prototype implementation. The experiment was performed on a SPARC Ultra in 2 seconds CPU time and 7.7 Mb of memory. We also tried to verify the protocol model without any substitutions or changing of bounds with our prototype, which did not terminate within 20 hours. Since we observed that the constraints lists of the states explored kept growing, we concluded that this experiment would not terminate at all. (Recall that parametric verification is undecidable.) The good news here is that in some cases,

the techniques for L/U automata yield results even if the state space exploration algorithm does not terminate on the original model.

The substitutions and techniques used in the verification to eliminate parameters are ad hoc. We believe however that more general strategies can be applied, especially in this case, where the constraints are L/U–like (i.e. can be written in the form $e \prec 0$ such that every p occurring negatively in e is a lower bound parameter and every p occurring positively in e is an upper bound parameter).

5 Experiments

5.1 A Prototype Extension of Uppaal

In this section, we report on the results of experimenting with a prototype extension of UPPAAL described in the previous sections.

Our prototype allows the user to give some initial constraints on the parameters. This is particularly useful when explorations cannot be finished due to lack of memory or time resources, or because a non-converging series of constraint sets is being generated. Often, partial results can be derived by observing the constraint sets that are generated during the exploration. Based on partial results, the actual solution constraints can be established in many cases. These partial results can then be checked by using an initial set of constraints. Always, for each parameter p the constraint $p \geq 0$ is added as initial constraint.

5.2 The Root Contention Protocol

The root contention protocol is part of a leader election protocol in the physical layer of the IEEE 1394 standard (FireWire/i-Link), which is used to break symmetry between two nodes contending to be the root of a tree, spanned in the network topology. The protocol consists of first drawing a random number (0 or 1), then waiting for some time according to the result drawn, followed by the sending of a message to the contending neighbor. This is repeated by both nodes until one of them receives a message before sending one, at which point the root is appointed.

We use the UPPAAL models of [18, 17], turn the constants used into parameters, and experiment with our prototype implementation (see Fig. 7 for results[3]). In both models, there are five constants, all of which are parameters in our experiments. The *delay* constant indicates the maximum

[3] All experiments were performed on a 366 MHz Celeron, except the liveness property which was performed in a 333 MHz SPARC Ultra Enterprise.

delay of signals sent between the two contending nodes. The *rc_fast_min* and *rc_fast_max* constants give the lower and upper bound to the waiting time of a node that has drawn 1. Similarly, the *rc_slow_min* and *rc_slow_max* constants give the bounds when 0 has been drawn. It is reasonable to assume that initially, the constraints $rc_fast_min \leq rc_fast_max \leq rc_slow_min \leq rc_slow_max$ hold.

We have checked for safety with the following property:

$$\forall\Box\ .\ (\neg(\mathrm{Node}_1.\mathrm{root} \wedge \mathrm{Node}_2.\mathrm{root}) \wedge \neg(\mathrm{Node}_1.\mathrm{child} \wedge \mathrm{Node}_2.\mathrm{child}))$$

Safety for [18] It is shown in [18], that the safety property holds, if the parameters obey the following relation: $delay < rc_fast_min$. We have checked that the error states, expressed in the safety property, are indeed unreachable when this parameter constraint is met. We have also checked whether error states are reachable when we assume the constraint $delay = rc_fast_min$. This turns out not to be the case. In fact, it is argued in Remark 2 in [18], that the mentioned constraint is not *needed* for the correctness of the protocol. Rather than checking this on the parameterized model without any initial constraints, which is a large task, we experiment with a non-parametric version of the model *without any timing constraints*. It turns out that this model satisfies the safety property, hence we deduce that the parametric model, in which guards and invariants have been added, satisfies the safety property for *any* valuation of the parameters.

Safety for [17] A different model of the root contention protocol is proposed in [17], in which it is shown that the relation between the parameters for the safety property to hold, should obey: $2{*}delay < rc_fast_min$. In fact, the model satisfies the safety property already when $delay < rc_fast_min$, but the stronger constraint is needed for proper behavior of the connecting wires. The necessity and sufficiency of these constraints is shown in [17] by applying standard UPPAAL to several valuations for the parameters, and presented as an experimental result.

We have checked that the error states, expressed in the safety property, are indeed unreachable when either of these parameter constraints are met. We have also checked whether error states are reachable when we assume the constraint $delay = rc_fast_min$, which turns out to be the case as well. In fact, the union of the constraint sets of reachable states reported, can be rewritten to the constraint $delay = rc_fast_min$. As a double-check, we have ascertained for some parameter valuations, satisfying $delay = rc_fast_min$, that standard UPPAAL also reaches an error state.

Since the model used for safety is a L/U automaton, we can experiment with Proposition 3, as follows. We show that our invariant property is satisfied by a more general model of root contention, and deduce with part 2 of Proposition 3 that it holds for the constraints we are after. We first identify the sets $L = \{rc_fast_min, rc_slow_min\}$ and $U = \{delay, rc_fast_max, rc_slow_max\}$. We substitute infinity for both *rc_fast_max* and *rc_slow_max*, *rc_fast_min* for *rc_slow_min*. The new model, together with either initial constraint $delay < rc_fast_min$, or $2{*}delay < rc_fast_min$, satisfies the invariant property. This allows us to conclude that the original model satisfies the invariant property for any valuation of the parameters where $rc_fast_min \leq rc_slow_min$, and the given initial constraint are satisfied. This includes the special case $rc_fast_min \leq rc_fast_max \leq rc_slow_min \leq rc_slow_max$.

We can do even better by applying Proposition 6, if we first change each guards or invariants for *delay* to a *strict* version, and then substitute infinity for both *rc_fast_max* and *rc_slow_max*, and *rc_fast_min* for both *delay* and *rc_slow_min*. Now we have a model with only one parameter and no constants, which we can verify non-parametrically with standard UPPAAL, for two valuations of the parameter *rc_fast_min*, namely 0 and a non-zero value. The invariant property is satisfied, hence, by Proposition 5, we can deduce that it holds for all valuations of *rc_fast_min*, hence the original model satisfies the invariant property for any valuation of the parameters where $rc_fast_min \leq rc_slow_min$, and $delay < rc_fast_min$. Likewise, we can substitute $rc_fast_min/2$ for *delay*, and derive the other constraint. As can be seen in Fig. 7, the speed-up in terms of memory and time is drastic.

Finally, we can combine the results for initial constraints $delay < rc_fast_min$ and $delay = rc_fast_min$ with the fact that our model is a L/U automaton, and derive the *necessity* of constraint $delay < rc_fast_min$, as follows. Suppose that a parameter valuation for *delay* and *rc_fast_min* exists, such that (1) the safety property holds, but (2) the constraint $delay < rc_fast_min$ is not satisfied. Assume this valuation assigns d to *delay* and r to *rc_fast_min*. By our results, we know that $d \neq r$, so $d > r$. We now apply Proposition 3, and deduce that for each parameter valuation that assigns a value to upper bound parameter *delay* which is *smaller* than d, and a value to lower bound parameter *rc_fast_min* which is *larger* than r, the safety property must hold. This includes valuations that satisfy constraint $delay = rc_fast_min$, which contradicts our results. We conclude that only for parameter valuations that satisfy constraint $delay < rc_fast_min$, the safety property holds.

model from	*initial constraints?*	*reduced?*	*property*	UPPAAL	*time*	*memory*
[18]	yes	no	safety	param	2.9 h	185 Mb
[18]	yes	yes	safety	std	1 s	800 Kb
[17]	yes	no	safety	param	1.6 m	36 Mb
[17]	yes	partly	safety	param	11 s	13 Mb
[17]	yes	completely	safety	std	1 s	800 Kb
[17]	yes	no	liveness	param	2.6 h	308 Mb

Fig. 7. Experimental results for the root contention protocol

Liveness for [17] In [17], it is also shown that a refinement relation between the model of the most detailed level, and a model which is a bit more abstract, holds when the following relations are obeyed: $2{*}delay < rc_fast_min$, and $2{*}delay < rc_slow_min$ - rc_fast_max. The refinement relation is such that it preserves both safety and liveness properties for the root contention protocol (which is proved in [17]). Again, the necessity and suffiency of the constraints is shown by experimenting with standard UPPAAL for several valuations for the parameters, and presented as an experimental result.

We have checked for a completely parameterized version of the system with the detailed model and the test automaton of the more abstract model, that error states in the test automaton are unreachable, that is, that the refinement relation holds. We have also checked, whether error states are reachable, that is, that the refinement relation does not hold, in the following two cases: either $2{*}delay = rc_fast_min$, and $2{*}delay < rc_slow_min$ - rc_fast_max, or $2{*}delay < rc_fast_min$, and $2{*}delay = rc_slow_min$ - rc_fast_max. This turns out to be the case as well. In fact, in both cases, the union of the constraint sets of reachable states reported, can be rewritten to these initial constraints. Again, this has been double checked by feeding parameter valuations that satisfy either of the above constraint sets to standard UPPAAL, which comes up with error states as well. Since the models for liveness use constraints that fall outside the scope of L/U automata, we cannot apply Proposition 6 here.

5.3 The Bounded Retransmission Protocol

This protocol was designed by Philips for communication between remote controls and audio/video/TV equipment. It is a slight alteration of the well-known alternating bit protocol, to which timing requirements and a bound on the retry mechanism have been added. In [8] constraints for the correctness of the protocol are derived by hand, and some instances

are checked using UPPAAL. Based on the models in [8], an automatic parametric analysis is performed in [4], however, no further results are given.

model from	*initial constraints*	*property*	UPPAAL	*time*	*memory*
[8]	yes	safety1	param	1.3 m	34 Mb
[8]	no	safety2	param	11 m	180 Mb
[8]	yes	safety2	param	3.5 m	64 Mb

Fig. 8. Experimental results for the bounded retransmission protocol

For our analysis we have also used the timed automata models from [8]. In [8] three different constraints are presented based on three properties which are needed to satisfy the safety specification of the protocol. We are only able to check two of these since one of the properties contain a parameter which our prototype version of UPPAAL is not able to handle yet.

One of the constraints derived in [8] is that $\mathrm{TR} \geq 2 \cdot \mathrm{MAX} \cdot \mathrm{T}_1 + 3 \cdot \mathrm{TD}$, where TR is the timeout of the receiver, T_1 is the timeout of the sender, MAX is the number of resends made by the sender, and TD is the delay of the channel. This constraint is needed to ensure that the receiver does not time out prematurely before the sender has decided to abort transmission. The sender has a parameter SYNC which decides for how long the sender waits until it expects that the receiver has realized a send error and reacted to it. In our parametric analysis we used TR and SYNC as parameters and instantiated the others to fixed values. Using our prototype we did derive the expected constraint $\mathrm{TR} \geq 2 \cdot \mathrm{MAX} \cdot \mathrm{T}_1 + 3 \cdot \mathrm{TD}$, however, we also derived the additional constraint $\mathrm{TR} - 2 \leq \mathrm{SYNC}$ which was not stated in [8] for this property. The necessity of this constraint was verified by trying models with different fixed values for the parameters. The full set of constraints derived in [8] includes a constraint $\mathrm{TR} \geq \mathrm{SYNC}$ which is based on the property we cannot check. Therefore the error we have encountered is only present in an intermediate result, the complete set of constraints derived is correct. The authors of [8] have acknowledged the error and provided an adjusted model of the protocol, for which the additional constraint is not necessary.

The last constraint derived in [8] arises from checking that the sender and receiver are not sending messages too fast for the channel to handle. In this model we treat T_1 as a parameter and derive the constraint $\mathrm{T}_1 > 2 \cdot \mathrm{TD}$ which is the same as is derived in [8].

5.4 Other Experiments

We have experimented with parameterized versions of models included in the standard UPPAAL distribution, namely Fischer's mutual exclusion protocol, a train gate controller, and a car gear box controller.

In the case of Fischer's protocol (which is the version of the standard UPPAAL distribution, and not the one discussed in the rest of this paper), we parameterized a model with two processes, by turning the bound on the period the processes wait, before entering the critical section, into a parameter. We were able to generate the constraints that ensure the mutual exclusion within 2 seconds of CPU time on a 266 MHz Pentium MMX. Using these constraints as initial constraints and checking that now indeed the mutual exclusion is guaranteed, is done even faster. Fischer's protocol with two processes was also checked in [4], which took about 3 minutes.

References

1. M. Abadi and L. Lamport. An old-fashioned recipe for real time. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings REX Workshop on Real-Time: Theory in Practice,* Mook, The Netherlands, June 1991, volume 600 of *Lecture Notes in Computer Science*, pages 1–27. Springer-Verlag, 1992.
2. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. In *Proceedings of the 25th Annual Symposium on Theory of Computing*, pages 592–601. ACM Press, 1993.
4. A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 419–434. Springer-Verlag, 2000.
5. G. Bandini, R. Lutje Spelberg, and H. Toetenel. Parametric verification of the IEEE 1394a root contention protocol using LPMC. http://tvs.twi.tudelft.nl/, July 2000. Submitted for publication.
6. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: A Model-Checking Tool for Real-Time Systems. In A.J. Hu and M.Y. Vardi, editors, *Proceedings of the 10th International Conference on Computer Aided Verification,* Vancouver, BC, Canada, volume 1427 of *Lecture Notes in Computer Science*, pages 546–550. Springer-Verlag, June/July 1998.
7. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* McGraw-Hill, Inc., 1991.
8. P.R. D'Argenio, J.-P. Katoen, T.C. Ruys, and J. Tretmans. The bounded retransmission protocol must be on time! In E. Brinksma, editor, *Proceedings of the Third Workshop on Tools and Algorithms for the Construction and Analysis of Systems,*

Enschede, The Netherlands, volume 1217 of *Lecture Notes in Computer Science*, pages 416–431. Springer-Verlag, April 1997.
9. D. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems,* Grenoble, France, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer-Verlag, 1990.
10. Ansgar Fehnker. Scheduling a Steel Plant with Timed Automata. In *Sixth International Conference on Real-Time Computing Systems and Applications (RTCSA'99)*. IEEE Computer Society Press, 1999.
11. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A Model Checker for Hybrid Systems. In O. Grumberg, editor, *Proceedings of the 9th International Conference on Computer Aided Verification*, volume 1254 of *Lecture Notes in Computer Science*, pages 460–463. Springer-Verlag, 1997.
12. IEEE Computer Society. IEEE Standard for a High Performance Serial Bus. Std 1394-1995, August 1996.
13. L. Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems*, 5(1):1–11, February 1987.
14. K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, October 1997.
15. R.F. Lutje Spelberg, W.J. Toetenel, and M. Ammerlaan. Partition refinement in real-time model checking. In A.P. Ravn and H. Rischel, editors, *Proceedings of the Fifth International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98),* Lyngby, Denmark, volume 1486 of *Lecture Notes in Computer Science*, pages 143–157. Springer-Verlag, 1998.
16. N.A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc., San Fransisco, California, 1996.
17. D.P.L. Simons and M.I.A. Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using Uppaal2k. Technical Report CSI-R0009, Computing Science Institute, University of Nijmegen, May 2000. Conditionally accepted for *Springer International Journal on Software Tools for Technology Transfer (STTT).*
18. M.I.A. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. In J.-P. Katoen, editor, *Proceedings 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems,* Bamberg, Germany, volume 1601 of *Lecture Notes in Computer Science*, pages 53–74. Springer-Verlag, 1999.
19. Available via `http://www.uppaal.com`.
20. S. Yovine. Model checking timed automata. In G. Rozenberg and F.W. Vaandrager, editors, *Lectures on Embedded Systems*, volume 1494 of *Lecture Notes in Computer Science*, pages 114–152. Springer-Verlag, October 1998.

A Notational Conventions

a	action
b	natural number
c	constraint
d	nonnegative real number
e	linear expression
f	simple guard

g guard
i, j index
k total number of actions
l lower bound parameter
m total number of clocks
n total number of parameters
p parameter
q location
r reset set
s state
t, T integer or real number
u upper bound parameter
v parameter valuation
w clock valuation
x, y clock
z parametric zone
A set of actions
C set of constraints
D parametric difference bound matrix
E set of linear expressions
G set of guards
I invariant function
K number of lower bound parameters
L set of lower bound parameters
M number of upper bound parameters
P set of parameters
Q set of locations
R set of reset sets
S set of states
U set of upper bound parameters
X set of clocks
$\mathcal{A}$ parametric timed automaton
E unit PDBM
$\mathcal{L}$ labelled transition system
N the natural numbers
R the real numbers
Z the integers
λ, μ extended valuation of lower bound (upper bound) parameter, respectively
ϕ state formula
ψ system property

Recent BRICS Report Series Publications

RS-01-5 **Thomas S. Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager.** ***Linear Parametric Model Checking of Timed Automata*****. January 2001. 44 pp. To appear in** ***Tools and Algorithms for The Construction and Analysis of Systems: 7th International Conference*****, TACAS '01 Proceedings, LNCS, 2001.**

RS-01-4 **Gerd Behrmann, Ansgar Fehnker, Thomas S. Hune, Kim G. Larsen, Paul Pettersson, and Judi Romijn.** ***Efficient Guiding Towards Cost-Optimality in*** **UPPAAL. January 2001. 21 pp. To appear in** ***Tools and Algorithms for The Construction and Analysis of Systems: 7th International Conference*****, TACAS '01 Proceedings, LNCS, 2001.**

RS-01-3 **Gerd Behrmann, Ansgar Fehnker, Thomas S. Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager.** ***Minimum-Cost Reachability for Priced Timed Automata*****. January 2001. 22 pp. To appear in** ***Hybrid Systems: Computation and Control*****, 2001.**

RS-01-2 **Rasmus Pagh and Jakob Pagter.** ***Optimal Time-Space Trade-Offs for Non-Comparison-Based Sorting*****. January 2001. ii+20 pp.**

RS-01-1 **Gerth Stølting Brodal, Anna Östlin, and Christian N. S. Pedersen.** ***The Complexity of Constructing Evolutionary Trees Using Experiments*****. 2001.**

RS-00-52 **Claude Crépeau, Frédéric Légaré, and Louis Salvail.** ***How to Convert a Flavor of Quantum Bit Commitment*****. December 2000. 24 pp. To appear in** ***Advances in Cryptology: International Conference on the Theory and Application of Cryptographic Techniques*****, EUROCRYPT '01 Proceedings, LNCS, 2001.**

RS-00-51 **Peter D. Mosses.** ***CASL for CafeOBJ Users*****. December 2000. 25 pp. Appears in Futatsugi, Nakagawa and Tamai, editors,** ***CAFE: An Industrial-Strength Algebraic Formal Method*****, 2000, chapter 6, pages 121–144.**

RS-00-50 **Peter D. Mosses.** ***Modularity in Meta-Languages*****. December 2000. 19 pp. Appears in** ***2nd Workshop on Logical Frameworks and Meta-Languages*****, LFM '00 Proceedings, 2000.**