Symbolic Techniques for Parametric Reasoning about Counter and Clock Systems

Aurore Annichini[1] Eugene Asarin[1] Ahmed Bouajjani[2]

[1] VERIMAG, Centre Equation, 2 av. de Vignate, 38610 Gières, France.
Aurore.Annichini@imag.fr, Eugene.Asarin@imag.fr

[2] LIAFA, Univ. of Paris 7, Case 7014, 2 place Jussieu, 75251 Paris Cedex 5, France.
abou@liafa.jussieu.fr

Abstract. We address the problem of automatic analysis of parametric counter and clock automata. We propose a semi-algorithmic approach based on using (1) expressive symbolic representation structures called Parametric DBM's, and (2) accurate extrapolation techniques allowing to speed up the reachability analysis and help its termination. The techniques we propose consist in guessing automatically the effect of iterating a control loop an arbitray number of times, and in checking that this guess is exact. Our approach can deal uniformly with systems that generate linear or nonlinear sets of configurations. We have implemented our techniques and experimented them on nontrivial examples such as a parametric timed version of the Bounded Retransmission Protocol.

1 Introduction

Counter automata and clock automata (timed automata) are widely used models of both hardware and software systems. A lot of effort has been devoted to the design of analysis techniques for these models (see e.g., [AD94, HNSY92, Hal93, BW94, BGL98, CJ98]). While the verification problem is undecidable in general for counter automata, this problem is decidable for timed automata [AD94], and there are model-checking algorithms and efficient verification tools for them [DOTY96, LPY97].

In this paper, we address the problem of analysing *parametric* counter and timed automata, i.e., models with counters and/or clocks that can be compared with parameters defined lower and upper bounds on their possible values. These parameters may range over infinite domains and are in general related by a set of constraints. We are interested in reasoning in a parametric way about the behaviours of a system: *verify* that the system satisfies some property for *all* possible values of the parameters, or *find* constraints on the parameters defining the set of all possible values for which the system satisfies a property. These two problems, i.e., parametric verification and parameter synthesis, can be solved (in the case of safety properties) as reachability problems in parametric models. Unfortunately, classical timed automata, where clocks can only be compared to constants, do not allow such a *parametric reasoning*. Moreover, it has been shown that for parametric timed automata, the reachability problem is undecidable [AHV93].

In this paper, we propose a semi-algorithmic approach that allows to deal with parametric counter and timed systems. We define new symbolic representations for use in their reachability analysis, and provide powerful and accurate techniques for computing representations of their sets of reachable configurations. The representation structures we define are extensions of the Difference Bound Matrices that are commonly used for representing reachability sets of (nonparametric) timed automata [Dil89, ACD+92, Yov98]. Our structures, called Parametric DBM's (PDBM's) encode constraints on counters and/or clocks expressing the fact that their values (and their differences) range in *parametric bound intervals*, i.e., the bounds of these intervals depend from the parameters. PDBM's are coupled with a set of constraints on the parameters. Such Constrained PDBM's allow to represent linear as well as nonlinear sets of configurations. We show in the paper how the basic manipulation operations on DBM's can be lifted to the parametric case, and then, we address the problem of computing the set of reachable configurations using Constrained PDBM's.

The main contribution of the paper is the definition of accurate extrapolation techniques that allow to speed up the computation of the reachability set and help the termination of the analysis. Our extrapolation technique consists in *guessing* automatically the effect of iterating a control loop (a loop in the control graph of the model) an arbitrary number of times, and *checking*, also automatically, that this guess is *exact*. More precisely, we can decide the exactness in the linear case and a subclass of the nonlinear case which can be reduced to the linear one. Hence, our extrapolation technique allows to generate automatically the *exact* set of reachable configurations. Furthermore, the extrapolation principle we propose is simple and uniform for counter and clock systems, which allows to consider systems with both kinds of variables. Another feature of our techniques is that they allow the automatic analysis of systems that generate *nonlinear* sets of configurations, which is beyond the scope of the existing algorithmic analysis techniques and tools.

We have implemented a package on Constrained PDBM's as well as a reachability analysis procedure based on our extrapolation techniques. We have experimented our prototype on nontrivial examples including systems generating linear sets of reachable configurations, as well as systems generating nonlinear sets of constraints. In all these examples, our analysis procedure terminates and generates the exact set of reachable configurations. These experiments show that our approach is powerful and accurate. In particular, we have been able to verify automatically a parametric timed version of the Bounded Retransmission Protocol (BRP) [HSV94]. The model we consider is a parametric timed counter system where parameters are constrained by nonlinear formulas (defined in [DKRT97]).

Outline: In Section 2, we give some basic definitions and introduce the kind of constraints and operations we use in our models. In Section 3, we introduce the parametric counter and timed systems. In Section 4, we introduce the PDBM's and the basic operations we consider on these structures. In Section 5, we define extrapolation techniques and show their use in reachability analysis. In Section 6, we discuss the current status of our implementation and experiments.

Related Work: The (semi-)algorithmic symbolic approach have been used for counter automata and timed systems in many works such as [CH78, HNSY92, Hal93, BW94, HHWT95, BGL98, BGP98, CJ98]. However, none of the existing works can deal with systems with nonlinear sets of reachable configurations.

Our extrapolation techniques have the same motivation as the widening operations [CH78, BGP98] used in the framework of abstract interpretation [CC77], and the techniques based on the use of meta-transitions [BW94, CJ98]. The aim of all these techniques is to speed up the computation of the reachable configurations and help the termination of the analysis. However, the existing widening techniques compute upper approximations using convex polyhedra. Our techniques are more accurate since they compute the exact effect of iterating an operation (control loop) an arbitrary number of times. Hence, our techniques are similar from this point of view to the techniques based on computing meta-transitions such as [BW94]. Compared with the technique of [BW94] for instance, our technique can sometimes detect "periodicities" more efficiently because it takes into account the set of configurations under consideration.

Furthermore, our extrapolation techniques are based on a principle of *guessing* the effect of the iterations which is in the same spirit as the principle of widening. But our principle differs technically from it and also differs from widening because we can check that our guess is exact. The principle of checking the exactness of a guess has been used in [FO97] in the context of system on strings. However, the techniques in [FO97] are different from ours, and [FO97] does not address the question of making a guess.

The problem of verifying the BRP has been addressed by several researchers [HSV94, GdP96, Mat96, DKRT97]. However these work either provide manual proofs, or use finite-state model-checking on abstract versions of the protocol or for particular instances of its parameters. In [AAB99a, AAB+99b] we have verified automatically an infinite-state version of the protocol with unbounded queues. However, we have considered in that work an abstraction of the clocks and counters and we have ignored the timing aspects that are addressed in this paper. In [DKRT97] the constraints that must be satisfied by the parameters are investigated. Then, their automatic verification using Uppaal is done only for a finite set of particular values satisfying these constraints. As far as we know, our work is the first one which allows to check automatically that these (nonlinear) constraints indeed allow the BRP to meet its specifications.

2 Preliminaries

Let $\mathcal{X}$ be a set of variables and let x range over $\mathcal{X}$. The set of *arithmetical terms* over $\mathcal{X}$, denoted $AT(\mathcal{X})$, is defined by the grammar:

$$t ::= 0 \mid 1 \mid x \mid t - t \mid t + t \mid t * t$$

The set of *first-order arithmetical formulas* over $\mathcal{X}$, denoted $FO(\mathcal{X})$, is defined by the grammar:

$$\phi ::= t \leq t \mid \neg\phi \mid \phi \vee \phi \mid \exists x.\ \phi \mid Is\_int(t)$$

Formulas are interpreted over the set of reals. The predicate Is_int expresses the constraint that a term has an integer value.

The fragment of $FO(\mathcal{X})$ of formulas without the Is_int predicate is called the *first-order arithmetics of reals* and denoted $RFO(\mathcal{X})$. The fragment of $FO(\mathcal{X})$ of formulas without multiplication ($*$) is called the *linear arithmetics* and is denoted $LFO(\mathcal{X})$. It is well-known that the problem of satisfiability in $FO(\mathcal{X})$ is undecidable, whereas it is decidable for both fragments $RFO(\mathcal{X})$ and $LFO(\mathcal{X})$.

Let $\mathcal{P}$ be a set of *parameters*. Then, a *simple parametric constraint* is a conjunction of formulas of the form $x \prec t$ or $x - y \prec t$, where $x, y \in \mathcal{X}$, $\prec \in \{<, \leq\}$, and $t \in AT(\mathcal{P})$. We denote by $SC(\mathcal{X}, \mathcal{P})$ the set of simple parametric constraints.

Each simple parametric constraint defines a family of convex polyhedra with parametric bounds. These polyhedra are of a special kind called *zones* in the timed automata literature. Notice that if all the bounds in a simple constraint are parameter-free terms (i.e., they represent constant values), then this constraint defines a unique convex polyhedron (zone).

We consider *simple operations* on variables corresponding to special kinds of assignments. We allow assignments of variables that are either of the form $x := y + t$ or of the form $x := t$, where $x, y \in \mathcal{X}$ are variables (x and y may be the same variable), and $t \in AT(\mathcal{P})$.

3 Parametric Timed Counter Systems

A Parametric Timed System (PTS) is a tuple $\mathcal{T} = (Q, C, P, I, \delta)$ where

- Q is a finite set of *control states*,
- $C = \{c_1, \ldots, c_n\}$ is a finite set of *clocks*,
- P is a finite set of *parameters*,
- $I : Q \to SC(C, P)$ is a function associating *invariants* with control states,
- δ is a finite set of *transitions* of the form (q_1, g, sop, q_2) where $q_1, q_2 \in Q$, $g \in SC(C, P)$ is a *guard*, and *sop* is a simple operation over C.

Clocks and parameters range over a set $\mathbb{D}$ which can be either the set of positive reals $\mathbb{R}^{\geq 0}$ (dense-time model) or the set of positive integers $\mathbb{N}$ (discrete-time model). Parameters can be seen as variables that are not modified by the system (they keep their initial values all the time). A configuration of $\mathcal{T}$ is a triplet $\langle q, \nu, \gamma \rangle$ where $q \in Q$, $\nu : C \to \mathbb{D}$ is a valuation of the clocks, and $\gamma : P \to \mathbb{D}$ is a valuation of the parameters.

Given a transition $\tau = (q_1, g, sop, q_2) \in \delta$, we define a transition relation $\to_\tau$ between configurations: $\langle q_1, \nu_1, \gamma_1 \rangle \to_\tau \langle q_2, \nu_2, \gamma_2 \rangle$ iff $(\nu_1, \gamma) \models g$ and $\nu_2 = sop(\nu_1)$. We also define a time-transition relation $\leadsto$ between configurations: $\langle q_1, \nu_1, \gamma_1 \rangle \leadsto \langle q_2, \nu_2, \gamma_2 \rangle$ iff $q_1 = q_2$ and $\exists r \in \mathbb{D}.\ \nu_2 = \nu_1 + r$ and $\forall r' \leq r.\ (\nu_1 + r', \gamma) \models I(q_1)$.

Let $\tau \in \delta$ and let Σ be set of configurations. Then, we define $post_\tau(\Sigma)$ to be the set $\{\sigma \ : \ \exists \sigma' \in \Sigma.\ \sigma' \rightsquigarrow \circ \rightarrow_\tau \circ \rightsquigarrow \sigma\}$, and $post(\Sigma) = \bigcup_{\tau \in \delta} post_\tau(\Sigma)$. Given a sequence of transitions $\theta = \tau_1, \ldots, \tau_n$, we define $post_\theta = post_{\tau_n} \circ \cdots \circ post_{\tau_1}$.

A Parametric Counter System (PCS) is a tuple $\mathcal{C} = (Q, X, P, \delta)$ where X is a set of integer valued variables (counters), and Q, P, and δ are defined in the same manner as for PTS's (substitute C by X in the definition of δ).

A configuration of $\mathcal{C}$ is a triplet $\langle q, \nu, \gamma \rangle$ where $q \in Q$, $\nu : X \rightarrow I\!\!N$, and $\gamma : P \rightarrow I\!\!N$. Given a transition $\tau \in \delta$, we define a relation $\rightarrow_\tau$ in the same manner as for PTS's. The function $post_\tau$ here is defined without considering time-transitions.

We can define also parametric models $\mathcal{M} = (Q, C, X, P, I, \delta)$ having both counters and clocks by a straightforward extension of the definitions of the PTS's and PCS's. We do not allow comparisons between clocks and counters in the guards and the invariants. We call these models Parametric Timed Counter Systems (PTCS's).

4 Symbolic representation structures

4.1 Parametric Difference Bound Matrices

To simplify the presentation, we consider here only the case of PTS's. The treatment of counters is analogous since we have the same kind of guards and operations on counters as on clocks. We introduce representation structures for sets of configurations of PTS's that are extensions of the Difference Bound Matrices used for representing reachability sets of (nonparametric) timed automata.

Let $\mathcal{T} = (Q, C, P, I, \delta)$ be a PTS, let $C = \{c_1, \ldots, c_n\}$ be the set its clocks, and let c_0 be an additional clock whose value is always equal to 0. Then, any simple parametric constraint can be represented by a $(n+1) \times (n+1)$ matrix M of elements in $AT(P) \times \{<, \leq\}$, where each entry $M(i,j) = (t, \prec)$ encodes the constraint $c_i - c_j \prec t$. We call such a matrix M a *Parametric Difference Bound Matrix* (PDBM).

A *parameter constraint* is a *quantifier-free* formula in $FO(P)$. A *Constrained PDBM* is a pair (M, Φ) where M is a PDBM and Φ is a parameter constraint. A *symbolic configuration* is a pair $(q, (M, \Phi))$ where $q \in Q$ is a control state, and (M, Φ) is a Constrained PDBM representing a set of clock and parameter values.

4.2 Basic Operations on Constrained PDBM's

We define operations for manipulating Constrained PDBM's by lifting all the standard operations on DBM's [Dil89, ACD+92, Yov98] to the parametric case. The operations that are worth discussing are: transformation into a canonical form (which is also used for emptyness check), intersection (used with guards and invariants when computing sets of successors), and inclusion test.

Canonical form Different constrained PDBM's can represent the same parametric set of configurations due to the fact that some of the bounds may not be tight enough. For instance, consider the constrained PDBM corresponding to the set of constraints:

$$S = (0 \leq x \leq p_2 \wedge 0 \leq y \leq p_1 \wedge 0 \leq x - y \leq 0, 0 \leq p_1 < p_2)$$

It can be seen that if the upper bound on x (i.e., p_2) is replaced by p_1, then we obtain another representation of the same parametric set of configurations as S. This representation corresponds to the *canonical form* of S.

We recall that in the nonparametric case, canonical forms of DBM's are constructed using the Floyd Warshall algorithm which computes the minimum path between all pairs of entries. In the parametric case we consider here, we follow the same principle by running a *symbolic* Floyd Warshall algorithm. During its execution, this algorithm needs to determine minimums between terms built from those appearing in the original matrix. (We omit here the technical discussion about how to deal with strict vs. nonstrict inequalities.) For that, the algorithm assumes each of the two possible cases and check their consistency w.r.t. the parameter constraints: given two terms t_1 and t_2, it considers the case where $min(t_1, t_2) = t_1$ (resp. $min(t_1, t_2) = t_2$) and adds $t_1 < t_2$ (resp. $t_1 \geq t_2$) in the parameter constraints, and then it delivers the consistent cases among these two, may be both of them. (We address below the decidability of this consistency check.)

For instance, the canonical form of S can be easily computed in this manner; the case splitting gives two cases but one of them is inconsistent ($p_1 \geq p_2$). However, if we remove $p_1 < p_2$ from S, we obtain two constrained PDBM's corresponding to each of the possible cases ($p_1 < p_2$ or $p_1 \geq p_2$). Notice that the construction of canonical forms allows also to test the emptyness of constrained PDBM's.

Now, in order to check the consistency of each of the possible cases when computing the minimum between two terms, we have to test the satisfiability of formulas ϕ of the form

$$\Phi(P) \wedge t_1 \prec t_2$$

where $\prec \in \{<, \leq\}$ and Φ is a parameter constraint. If Φ is in $LFO(P)$ (linear constraint) or in $RFO(P)$ (all parameters are reals), then this test is decidable.

If Φ is a nonlinear formula of $FO(P)$ mixing integer and real parameters, this test is of course undecidable. Nevertheless, we still can test *safely* the satisfiability of ϕ in $RFO(P)$ (i.e., we check the satisfiability of ϕ under the assumption that all the parameters are reals). If ϕ in not satisfiable in $RFO(P)$, we are sure that it is not satisfiable for its original interpretation in $FO(P)$. However, ϕ could be satisfiable in $RFO(P)$ whereas there are no integer valuations of P satisfying it. Hence, by interpreting formulas of $FO(P)$ in $RFO(P)$, we consider *upper approximations* of the sets of possible configurations.

Intersection Given two constrained PDBM's $S_1 = (M_1, \Phi_1)$ and $S_2 = (M_2, \Phi_2)$, the intersection of S_1 and S_2 is represented in general by a *set* of constrained PDBM's. Roughly speaking, the construction consists in computing for every i and j, the minimum between the two *terms* $M_1(i,j)$ and $M_2(i,j)$, under the parameter constraints $\Phi_1 \wedge \Phi_2$. This is done by case splitting and checking the consistency of each case, as in the construction of canonical representations explained above.

Again, checking the consistency of the different cases produced by case splitting is decidable if the parameter constraints are in $LFO(P)$ or $RFO(P)$. Hence, in this case the construction of the intersection is exact. In the general case, checking satisfiability in $RFO(P)$ instead of $FO(P)$ is a safe consistency test that yields an upper approximation of the intersection.

Test of inclusion Let $S_1 = (M_1, \Phi_1)$ and $S_2 = (M_2, \Phi_2)$ be two canonical constrained PDBM's. The inclusion of S_1 in S_2 can be expressed by the following formula ψ:

$$\forall P.\ \Phi_1(P) \wedge \Phi(P) \Rightarrow M_1 \leq M_2$$

The validity of ψ is decidable if it is in $LFO(P)$ or in $RFO(P)$. Otherwise, we have a safe test of inclusion by checking the validity of ψ in $RFO(P)$. Indeed, if ψ is valid $RFO(P)$, then it is also valid in $FO(P)$ and hence, if our inclusion test answer positively, we are sure that it is true. However, if ψ is not valid, it does not mean that S_1 is not included in S_2.

5 Reachability analysis

5.1 Building Symbolic Reachability Graphs

Let $\mathcal{T}$ be a PTCS. We present a procedure which, given a symbolic configuration S, computes a representation of the set $post^*(S)$. For that, starting from S, we construct a symbolic reachability graph where each vertex is a symbolic configuration and edges correspond to transitions of $\mathcal{T}$. The vertices of the symbolic graph are treated according to a depth-first traversal. The construction stops when each symbolic configuration that can be generated is covered by (included in) some symbolic configuration that has been already computed. During this construction, we use extrapolation in order to help termination.

Our extrapolation technique is based on guessing automatically the effect of iterating an arbitrary number of times a control loop (cycle in the control graph of $\mathcal{T}$), starting from a given symbolic configuration, and checking that this guess is exact (does not introduce nonreachable configuarations). Informally, we can present our extrapolation principle as follows: Let S be a symbolic constraint and let θ be a control loop, and suppose that the difference (in a sense which will be defined later) between $post_\theta(S)$ and S, say Δ, is equal to the difference between $post^2_\theta(S)$ and $post_\theta(S)$. Then, we suspect that the effect of iterating θ will be to add at each step the same Δ to the original set, i.e., after n iterations, the set of reachable configurations will be roughly $S + n\Delta$ (the precise set is given below). Roughly speaking, our technique consists in guessing that a control loop θ (which is may be a composition of several simple loops) defines a periodic operation starting from a *particular* set of configurations. In many cases, our guess is exact. Moreover, the exactness of the effect obtained by extrapolation can be expressed as an arithmetical formula (we discuss later the decidability issue of this check).

Notice that our extrapolation technique introduces new parameters (n) corresponding to numbers of iterations of control loops. In order to deal with sets represented by means of such variables, we have to extend our symbolic representations and introduce *open* constrained PDBM's.

Let us call *iteration parameters* these auxiliary variables and let N be the set of such variables. In order to deal with sets represented using iteration variables we have to extend our symbolic representation and introduce *open* PDBM's.

5.2 Open Constrained PDBM's

Let N be a (countable) set of *iteration variables*. An *open* PDBM is a PDBM such that its elements are terms are in $AT(P \cup N) \times \{<, \leq\}$. We extend also the definitions of Constrained PDBM's and symbolic configurations by considering that the terms appearing in parameter constraints are in $AT(P \cup N)$.

Now, let us see how to extend the operations on PDBM to open PDBM's. The construction of canonical forms as well as intersection can be done as previously. The problematic operation if the test of inclusion. Indeed, given two canonical constrained open PDBM's $S_1 = (M_1, \Phi_1)$ and $S_2 = (M_2, \Phi_2)$, the inclusion of S_1 in S_2 can be expressed by the formula ψ:

$$\forall P.\, \forall N.\ (Is\_int(N) \wedge \Phi_1(P, N) \Rightarrow \exists N'.\ Is\_int(N') \wedge \Phi_2(P, N') \wedge M_1(N) \leq M_2(N')$$

When ψ is a linear formula (a LFO formula), the validity of ψ is decidable, and hence, the inclusion problem between constrained open PDBM's is decidable in this case.

Another interesting case is what we call the *half-linear case* which corresponds to the following situation: using quantifier elimination, the obtained formula from ψ after eliminating all the *real valued* parameters in P is a *linear* formula on $N \cup N'$. The validity of this formula can be checked since LFO on integers (Presburger arithmetics) is decidable. The elimination of the real valued parameters can be done automatically using the techniques of quantifier elimination in RFO (we do not need to assume that N and N' are sets of integer variables).

Using this technique, we can deal with significant cases of systems generating nonlinear sets of configurations. For instance, in the analysis of the Bounded Retransmission Protocol, all the inclusion tests are half-linear.

Beyond the class of half-linear systems, the test of inclusion is undecidable. Nevertheless, even in this general case, it is possible to have a *safe* test of inclusion. However, we cannot adopt the naive approach which consists in checking the validity of ψ in RFO since ψ has an alternation of universal and existential quantification. Then, the solution we propose is to define a formula ψ' in RFO which is "reasonably" stronger than ψ. This formula is:

$$\forall P.\, \forall N.\, \Phi_1(P,N) \Rightarrow \exists N'.\, \forall N''.\, \bigwedge_{i=1}^{|N'|} |N'_i - N''_i| \leq \frac{1}{2} \Rightarrow \Phi_2(P,N'') \wedge M_1(N) \leq M_2(N'')$$

The idea is to require that for every real vector N, there is a real vector N' such that $M_1 \leq M_2$ holds for all the real vectors N'' in a neighborhood of N' which contains at least one integer vector. Thus, if ψ' holds, necessarily ψ holds too.

5.3 Extrapolation

We present hereafter our extrapolation principle. We need first to introduce some notations.

Let $\mathcal{T} = (Q, C, X, P, I, \delta)$ be a PTCS. A *control loop* is a cycle in the graph (Q, δ), i.e., a path $(q_1, g_1, sop_1, q'_1) \ldots (q_n, g_n, sop_n, q'_n)$ such that $q_1 = q'_n$ and $\forall i \in \{1, \ldots, n-1\}$, $q'_i = q_{i+1}$.

Given an open PDBM M (resp. symbolic constraint $S = (M, \Phi)$), we denote by $Iter(M)$ (resp. $Iter(S)$) the set of iteration variables appearing in M (resp. in M or Φ). Let $S = (M, \Phi)$ be a constrained PDBM. Given $n \in N$ such that $n \notin Iter(S)$, we denote by $S{\uparrow_n}$ the constrained PDBM $(M, \Phi \wedge n \geq 0)$. Given a PDBM M' such that $Iter(M') \subseteq Iter(S)$, we denote by $S + M'$ the symbolic constraint $(M + M', \Phi)$.

Now, let θ be a control loop and let (q, S) be a symbolic configuration, where $S = (M, \Phi)$ is a constrained PDBM. Then, suppose we have computed $S_1 = (M_1, \Phi_1)$ and $S_2 = (M_2, \Phi_2)$ such that $(q, S_1) = post_\theta(q, S)$ and $(q, S_2) = post_\theta(q, S_1)$. Let $\Delta = M_1 - M$ and $\Delta' = M_2 - M_1$. Our extrapolation principle consists in checking whether the two following conditions hold:

C1: $\forall P.\, \forall N.\, Is\_int(N) \wedge \Phi_2(P,N) \Rightarrow \Delta = \Delta'$,
C2: $\forall n \geq 0.\, post^2_\theta(q, S{\uparrow_n} + n\Delta) = post_\theta(q, S{\uparrow_n} + (n+1)\Delta)$,

and, if C1 and C2 hold, in adding $post_\theta(q, S{\uparrow_n} + n\Delta)$ to the computed set of reachable configurations, and the edge $(q, S_1) \xrightarrow{\theta^*} post_\theta(q, S{\uparrow_n} + n\Delta)$ to the symbolic graph.

Condition C1 says that the effect of θ after two iterations is to add Δ at each step. Notice that we check the equality of the two matrices $M_2 - M_1$ and $M_1 - M$ under the constraint Φ_2. This constraint is stronger than Φ_1 which is itself stronger than Φ due to the fact that each application of θ may introduce but never remove parameter constraints. Condition C2 allows to check that each application of θ has an effect of adding Δ, provided the guards and the invariants in θ are satisfied. In order to take into account the guards and invariants, we compute the effect after $n+1$ iterations of θ as the $post_\theta$-image of $(q, S{\uparrow_n} + n\Delta)$. We can prove, by straightforward inductions, the following fact:

Lemma 5.1 *Let θ be a control loop, let (q, S) be a symbolic configuration, and let Δ be a PDBM. Then, the two following formulas are equivalent:*

1. $\forall n \geq 0.\ post_\theta^2(q, S{\uparrow}_n + n\Delta) = post_\theta(q, S{\uparrow}_n + (n+1)\Delta)$,
2. $\forall n \geq 0.\ post_\theta^{n+1}(q, S) = post_\theta(q, S{\uparrow}_n + n\Delta)$.

By Lemma 5.1, we can deduce that when it can be applyed, our extrapolation principle is exact (it computes precisely the set $post_\theta^{n+1}(q, S)$, for any $n \geq 0$).

Both conditions C1 and C2 correspond to arithmetical formulas. These conditions are of course decidable in the linear case, and they are also decidable in the half-linear case, i.e., after elimination of the real-valued parameters in P, the obtained formula is linear. Hence, we have an *exact* extrapolation technique in these case.

In the general (nonlinear) case, the test of exactness C2 is actually not relevant, since we can only compute upper-approximations of the reachability set in this case. So, the extrapolation principle we apply in this case is a *weak extrapolation principle* which consists in checking condition C1 only.

Actually, even in the linear and half-linear cases, it is often not necessary to check the condition C2. It can be observed that the *weak extrapolation principle*, even if it not guaranteed to compute the exact reachability set (it computes and upper-approximation of it in general), it is more accurate than existing widening operators [CH78, Hal93] since it allows to capture periodicities (see the examples in Section 5.1). We have used this principle to analyse several examples of parametric counter and timed systems and in all these cases, our procedure was able to compute the *exact* set of reachable configurations. In fact, we can prove that for an important class of systems, the weak extrapolation principle is exact. This class includes many of the usual examples encountered in the literature (Bakery algorithm, lift controler, etc). For lack of space, we omit addressing this issue in this version of the paper.

5.4 Examples

Let us illustrate the use of our reachability analysis techniques on small examples.

A simple linear system: Let us consider first a very simple *counter* system which is described in Figure 1. In this example, x is counter and T is a parameter.

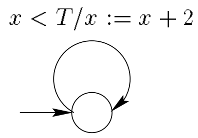

Fig. 1. Linear counter system

We suppose that the initial value of x is 0 and that T is not contrained. So, the initial symbolic configuration is $(0, true)$. The first execution of the unique transition θ of the system creates the edge:

$$(0, true) \xrightarrow{\theta} (2, 0 < T)$$

since x is incremented by 2, and before that (when its value was 0), it was compared to T. The second iteration creates the edge

$$(2, 0 < T) \xrightarrow{\theta} (4, 2 < T)$$

At this point, we can check that condition C1 holds since the effect of the first and second iteration of θ is to add the same value 2 to x. Then, by the weak extrapolation principle, the following edge is created

$$(2, 0 < T) \xrightarrow{\theta^*} post_\theta((0, 0 \leq n)+2n) = post_\theta(2n, 0 \leq n) = (2n+2, 0 \leq n \wedge 2n < T)$$

Notice that the application of θ to the symbolic configuration $(2n, n \geq 0)$ allowed to generate the constraint $2n < T$ relating n with T. This illustrates how guards (and also invariants in the case of timed systems) are taken into account in the extrapolation technique.

It can be easily checked that condition C2 also holds. Indeed, we have

$$post^2_\theta(2n, 0 \leq n) = (2n + 4, 0 \leq n \wedge 2n + 2 < T) = post_\theta(2n + 2, 0 \leq n)$$

and thus, we are sure that our extrapolation is exact.

It worths noting that our extrapolation principle can generate periodic sets whereas classical widening techniques [CH78, Hal93] will not.

A simple half-linear system: Now, let us consider the parametric timed automaton given in Figure 2, where c and c' are real-valued clocks, and T and T' are real-valued parameters. Let us assume that the initial configuration is

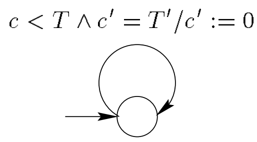

Fig. 2. Half-linear clock system

$(c = 0, c' = 0, 0 \leq T \wedge 0 \leq T')$. Then, the first application of the transition θ of the system gives:

$$(0, 0, 0 \leq T \wedge 0 \leq T') \xrightarrow{\theta} (T', 0, 0 < T \wedge 0 \leq T')$$

and the second iteration gives:

$$(T', 0, 0 < T \wedge 0 \leq T') \xrightarrow{\theta} (2T', 0, 0 \leq T' < T)$$

Then, the condition C1 of our extrapolation principle holds with $\Delta(c) = T'$ and $\Delta(c') = 0$. Following the weak extrapolation principle, we create a θ^*-edge from $(T', 0, 0 < T \wedge 0 \leq T')$ to $post_\theta(n * T', 0, 0 \leq n \wedge 0 \leq T \wedge 0 \leq T')$ wich is equal to:

$$((n + 1) * T', 0, 0 \leq n \wedge 0 \leq T' \wedge n * T' < T)$$

The generated symbolic configuration is nonlinear, but still, all decision problems we need on it are decidable due to the fact that they are half-linear. For instance, after generating this symbolic configuration, we need to check for its emptyness by deciding whether the parameter constraints are satisfiable. This is very simple in this example because after eliminating the parameters T and T' (supposed to be real valued), we get a trivial linear constraint on n which is $n \geq 0$. It can also be seen that in this case, the condition C2 (exactness of the extrapolation) can be done straigthforwardly.

A complex half-linear system: We consider here an example which is inspired from the model of the Bounded Retransmission Protocol. It consists of a systems with two clocks c_1 and c_2 and a counter x (see Figure 3). Intuitively, c_1 represents

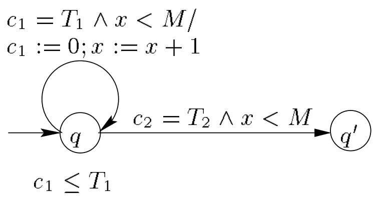

Fig. 3. The Nonlinear Kernel of the BRP

the clock of a sender and c_2 the clock of a receiver. These clocks are compared with parametric bounds T_1 and T_2 supposed to be real values. The counter x counts the number of times the loop θ on the state q is performed. The transition θ corresponds in the BRP to a retransmission action by the sender. The number of these retransmissions is bounded by M which is an integer parameter. We assume that the parameters T_1, T_2, and M are related by a nonlinear constraint

$$T_2 > M * T_1$$

which means that the timeout of the receiver is at least M (the maximum number of retransmissions) times the timeout of the sender. The question is whether the state q_1 (considered as a bad state) is reachable under the constraint above.

Roughly speaking, this correponds to a property of synchronisation between the sender and receiver in the BRP: The timeout of the receiver should not expire before the sender has finished all his possible retransmissions. Notice that the constraint above is not precisely the one considered on the timeouts in the BRP [DKRT97], but our aim here is just to show simply and faitfuly the main problems that appear in the analysis of a complex system such as the BRP.

Let us compute the reachability set starting from the initial configuration:

$$0 \leq c_1 \leq T_1 \wedge 0 \leq c_2 \leq T_1 \wedge c_1 - c_2 = 0 \wedge x = 0,$$
$$0 \leq T_1 \wedge 0 \leq T_2 \wedge 0 \leq M \wedge T_2 > M * T_1$$

The two first applications of θ give successively

$$0 \leq c_1 \leq T_1 \wedge T_1 \leq c_2 \leq 2T_1 \wedge c_1 - c_2 = T_1 \wedge x = 1,$$
$$0 \leq T_1 \wedge 0 \leq T_2 \wedge 0 < M \wedge T_2 > M * T_1$$

and

$$0 \leq c_1 \leq T_1 \wedge 2T_1 \leq c_2 \leq 3T_1 \wedge c_1 - c_2 = 2T_1 \wedge x = 2,$$
$$0 \leq T_1 \wedge 0 \leq T_2 \wedge 1 < M \wedge T_2 > M * T_1$$

At this point, it can be checked that the extrapolation condition C1 holds with $\Delta(c_1) = 0$, $\Delta(c_2) = T_1$, $\Delta(c_1 - c_2) = T_1$, and $\Delta(x) = 1$. (Notice that, in general, the differences between lowers bounds and upper bounds could be different, and thus, Δ may correspond to nondeterministic increasings represented by intervals instead of precise values as in this example and the previous ones.) Then, by the weak extrapolation principle we can create the configuration:

$$0 \leq c_1 \leq T_1 \wedge (n+1) * T_1 \leq c_2 \leq (n+2) * T_1 \wedge c_1 - c_2 = (n+1) * T_1 \wedge x = n+1,$$
$$0 \leq n \wedge 0 \leq T_1 \wedge 0 \leq T_2 \wedge n < M \wedge T_2 > M * T_1$$

The transition to the state q_1 is executable if the following set of parameter constraints (obtained by intersection with the guard) is satisfiable:

$$0 \leq n \wedge 0 \leq T_1 \wedge 0 \leq T_2 \wedge n+1 < M \wedge (n+1) * T_1 \leq T_2 \leq (n+2) * T_1 \wedge T_2 > M * T_1$$

The formula above is actually half-linear because after the elimination of the real parameter T_2 we obtain the constraint

$$0 \leq n \wedge 0 \leq T_1 \wedge n + 1 < M \wedge (n+2) * T_1 > M * T_1$$

and after the elimination of T_1 we obtain the formula:

$$0 \leq n \wedge n + 1 < M \wedge M < n + 2$$

This formula is a linear constraint on integer variables, and thus its satisfiability can be decided. Clearly the formula above is unsatisfiable since there is no integers strictly between two successive integers. Hence, we conclude that q_1 is not reachable.

Notice that we have omitted here the test of condition C2. Actually, this test can also be decided as a half-linear satisfiability problem.

6 Implementation and Experiments

We have implemented a package of Constrained PDBM's containing all the basic manipulation operations. Our current implementation uses the tool REDLOG/REDUCE [DS97, DSW98] for quantification elimination and deciding satisfiability in *RFO*, and uses the tool OMEGA [BGP97] for deciding satisfiability in Presburger arithmetics. In the nonparametric case, our package behaves as a DBM package: all operations involving only constants are done without case splitting (used for comparing parametric terms) and without invoking REDLOG or OMEGA).

Based on this package, we have implemented a procedure for reachability analysis using extrapolation. This procedure computes, when it terminates, a symbolic graph of a given PTCS starting from a given symbolic configuration. We have applied our procedure to several examples including counter and timed systems that generate linear constraints such as:

- the Bakery algorithm for mutual exclusion with unbounded ticket counters (0.82 sec with Omega as constraint solver) and two processes,
- the timed parametric Fisher's mutual exclusion protocol (169.64 sec., Omega) with two processes,
- the parametric lift controler of [Val89] where the number of floors is a parameter (286.32 sec, Omega),

as well as complex systems generating *nonlinear constraints* relating clocks and counters. Indeed, we have applied our techniques to analyse the Bounded Retransmission Protocol which involves a nontrivial parametric reasoning on both counters and clocks ($\sim$ 91 mn, Redlog/Reduce). We have considered for this example the modelisation given in [DKRT97]. Our reachability analysis procedure has been able to construct a symbolic graph corresponding to the partition of the set of reachable configurations according to control states. This symbolic graph represents a finite abstraction of the original infinite-state model. After projection on external actions and minimisation (using the CADP toolbox [FGK+96]), we got a finite model with 7 states on which the safety properties of the BRP has been automatically checked.

The analysis of linear systems such as the Bakery algorithm and the lift controler has been already done by other researchers using different techniques such as widening [BGP97, BGL98] or the computation of meta-transitions [BW94]. Our experiments show that our techniques are powerful enough to deal with all these cases as well as, and in a uniform way, with systems generating nonlinear constraints that are beyond the scope of the existing methods and tools. In all the cases we considered, our procedure was able to compute the *exact* set of reachable configurations.

7 Conclusion

We have introduced an extrapolation principle for analysing systems with counters and clocks based on the use of Parametric DBM's. Our approach is an extension of the existing methods on timed automata to the case of systems with parametric constraints. An interesting feature of our techniques is that they can be applied uniformly to nonparametric or to parametric systems, to linear systems or to nonlinear ones (which are beyond the scope of the known techniques and tools). Moreover, our techniques are accurate and generate exact reachability sets for a wide class of systems.

We have implemented our techniques in a tool prototype using Redlog/Reduce and Omega as constraint solvers. The experiments we have done with our prototype show that our approach is powerful and effective. In particular, we have been able to verify automatically a parametric timed version of the Bounded Retransmission Protocol. Future work includes studying other symbolic representations and associated extrapolation techniques, and identifying classes of arithmetical constraints that can be handled efficiently. In particular, it would be interesting to investigate parametric extensions of structures like CDD's [LPWY99] and DDD's [MLAH99].

Finally, let us mention that in this paper we have addressed only *forward* reachability analysis. Actually, the techniques we have developed can also be used for *backward* analysis as well.

References

[AAB99a] P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic Verification of Lossy Channel Systems: Application to the Bounded Retransmission Protocol. In *TACAS'99*. LNCS 1579, 1999.

[AAB+99b] P.A. Abdulla, A. Annichini, S. Bensalem, A. Bouajjani, and Y. Lakhnech P. Habermehl. Verification of Infinite-State Systems by Combining Abstraction and Reachability Analysis. In *11th Intern. Conf. on Computer Aided Verification (CAV'99)*. LNCS 1633, 1999.

[ACD+92] R. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, and H. Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *RTSS'92*. IEEE, 1992.

[AD94] R. Alur and D. Dill. A Theory of Timed Automata. *TCS*, 126, 1994.

[AHV93] 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.

[BGL98] Tevfik Bultan, Richard Gerber, and Christopher League. Verifying systems with integer constraints and Boolean predicates: A composite approach. In *Proc. of the Intern. Symp. on Software Testing and Analysis*. ACM Press, 1998.

[BGP97] T. Bultan, R. Gerber, and W. Pugh. Symbolic model checking of infinite state systems using Presburger arithmetic. *LNCS*, 1254, 1997.

[BGP98] Tevfik Bultan, Richard Gerber, and William Pugh. Model checking concurrent systems with unbounded integer variables: Symbolic representations, approximations and experimental results. Tech. Rep. CS-TR-3870, University of Maryland, College Park, 1998.

[BW94] B. Boigelot and P. Wolper. Symbolic Verification with Periodic Sets. *Lecture Notes in Computer Science*, 818:55–67, 1994.

[CC77] P. Cousot and R. Cousot. Static Determination of Dynamic Properties of Recursive Procedures. In *IFIP Conf. on Formal Description of Programming Concepts*. North-Holland Pub., 1977.

[CH78] Patrick Cousot and Nicholas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL'78*. ACM, 1978.

[CJ98] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In *CAV'98*. LNCS 1427, 1998.

[Dil89] D. Dill. Timing assumptions and verification of finite-state concurrent systems. In *CAV'89*. LNCS 407, 1989.

[DKRT97] P. D'Argenio, J-P. Katoen, T. Ruys, and G.J. Tretmans. The Bounded Retransmission Protocol must be on Time. In *TACAS'97*. LNCS 1217, 1997.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The Tool KRONOS. In *Hybrid Systems III, Verification and Control*. LNCS 1066, 1996.

[DS97] A. Dolzmann and T. Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, 1997.

[DSW98] A. Dolzmann, T. Sturm, and V. Weispfenning. A new approach for automatic theorem proving in real geometry. *Automated Reasoning*, 21(3):357–380, 1998.

[FGK+96] J-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP: A Protocol Validation and Verification Toolbox. In *CAV'96*. LNCS 1102, 1996.

[FO97] L. Fribourg and H. Olsen. Reachability sets of parametrized rings as regular languages. In *Infinity'97*. volume 9 of Electronical Notes in Theoretical Computer Science. Elsevier Science, 1997.

[GdP96] J-F. Groote and J. Van de Pol. A Bounded Retransmission Protocol for Large Data Packets. In *AMAST'96*. LNCS 1101, 1996.

[Hal93] N. Halbwachs. Delay Analysis in Synchronous Programs. In *CAV'93*. LNCS 697, 1993.

[HHWT95] T. Henzinger, P.H. Ho, and H. Wong-Toi. A User Guide to HyTech. In *TACAS'95*. LNCS 1019, 1995.

[HNSY92] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic Model-Checking for Real-Time Systems. In *LICS'92*. IEEE, 1992.

[HSV94] L. Helmink, M.P.A. Sellink, and F. Vaandrager. Proof checking a Data Link Protocol. In *Types for Proofs and Programs*. LNCS 806, 1994.

[LPWY99] K. Larsen, J. Pearson, C. Weise, and W. Yi. Efficient timed reachability analysis using clock difference diagrams. In *CAV'99*. LNCS 1633, 1999.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi. UPPAAL: Status and Developments. In *CAV'97*. LNCS, 1997.

[Mat96] R. Mateescu. Formal Description and Analysis of a Bounded Retransmission Protocol. Technical report no. 2965, INRIA, 1996.

[MLAH99] J. Moller, J. Lichtenberg, H.R. Andersen, and H. Hulgaard. Difference decision diagrams. Tech. rep it-tr-1999-023, Department of Information Technology, Technical University of Denmark, 1999.

[Val89] A. Valmari. State generation with induction. In *Scandinavian Conference on Artificial Intelligence*, 1989.

[Yov98] S. Yovine. Model-checking timed automata. *Embedded Systems, LNCS*, 1998.

This article was processed using the LaTeX macro package with LLNCS style