Analyzing Fair Parametric Extended Automata

A. Bouajjani[2], A. Collomb-Annichini[1], Y. Lakhnech[1], and M. Sighireanu[2]

[1] Verimag, Centre Equation, 2 av. de Vignate, 38610 Gières, France
Aurore.Annichini@imag.fr, Yassine.Lakhnech@imag.fr
[2] Liafa, University of Paris 7, 2 place Jussieu, 75251 Paris cedex 5, France
abou@liafa.jussieu.fr, sighirea@liafa.jussieu.fr

Abstract. We address the problem of verifying safety and liveness properties for infinite-state systems, using symbolic reachability analysis. The models we consider are *fair parametric extended automata*, i.e., counter automata with parametric guards, supplied with fairness conditions on their transitions. In previous work, we shown that symbolic reachability analysis using acceleration techniques can be used to generate finite abstractions (symbolic graphs) of the original infinite-state model. In this paper, we show that this analysis can be also used to introduce fairness conditions on the generated abstract model allowing to model-check liveness properties. We show first how to translate faithfully the fairness conditions of the infinite-state original model to conditions on the generated finite symbolic graph. Then, we show that we can also synthesize automatically new fairness conditions allowing to eliminate infinite paths in the symbolic graph which do not correspond to valid behaviours in the original model. These infinite paths correspond to abstractions of boundedly iterable (nested) loops. We show techniques allowing to decide this bounded iterability for a class of components in the symbolic graph. We illustrate the application of these techniques to nontrivial examples.

1 Introduction

Symbolic reachability analysis is a powerful paradigm used in the verification of infinite-state systems, such as extended automata, communicating automata through unbounded queues, parameterized systems, etc. It consists in using finite structures to represent infinite sets of configurations, and iterative exploration procedures to compute (a finite representation of) the set of all reachable configurations, or an upper approximation of this set. To help termination, and in some cases to force it, these procedures are often enhanced by acceleration techniques which allow to compute in one step the effect of sequences of transitions instead of one single transition in the system. Typically, an acceleration step corresponds to the computation of (an upper approximation of) the set of reachable configurations by iterating an arbitrary number of times some sequence of transitions (a circuit in the control graph of the system). For instance, starting from an initial value $x = 0$, the iteration of a transition which increments the variable x by 2 leads to the set of configurations $\{0, 2, 4, \ldots\}$ which can be represented by the constraint $x = 2n$, with $n \geq 0$. Acceleration techniques allow to compute this finite representation in one step instead of computing the infinite sequence of approximations $\{0\}, \{0,2\}, \{0,2,4\}, \ldots$

The construction of the set of reachable configurations allows to check (on-the-fly) safety properties. More interestingly, this construction can also be used to generate finite abstractions of the analyzed model on which standard finite model-checking procedures can be applied. Indeed, given any finite partition of the set of configurations, it is possible to construct a finite symbolic graph (the quotient graph according to this partition) which simulates the original system. In general, the exploration procedures used to generate the reachability set produce a finite symbolic graph, when they terminate. The nodes of this graph are symbolic configurations (sets of configurations) and the edges correspond to the application of transitions of the system or acceleration steps (meta-transitions). Since the symbolic graph simulates the analyzed system, if it satisfies a property then the original system also satisfies it (we suppose here that properties involve only universal path quantification, e.g., linear-time properties).

In previous works, we have shown that this approach can be applied to verify fully automatically safety properties of infinite-state models such as communicating automata with unbounded queues [1], and parametric extended automata (automata with clocks and counters) [5]. However, while very successful in the case of safety properties, this approach fails in general when applied to the verification of liveness properties. This is due to the fact that, almost always, liveness properties only hold under fairness conditions saying that infinite iterations of certain transitions are not allowed. Therefore, the question we address is, given an infinite-state model with fairness conditions, how to introduce automatically fairness conditions on the finite abstract model generated by means of symbolic reachability analysis, in order to be able to model-check liveness properties.

In fact, the fairness conditions that must be introduced on the symbolic graph are of two sorts. First, the original infinite-state model may have fairness conditions expressed on its transitions (in its infinite configuration graph). These conditions must be translated faithfully on the transitions of the finite abstract model (symbolic configuration graph). Moreover, additional fairness conditions must be inferred since it is often the case that the finite abstract model has loops that do not correspond to a possible infinite execution path in the original model. For instance, consider a transition θ which increments a variable x by 2 if $x \leq M$, where M is a positive integer parameter. The set of reachable configurations from the initial value $x = 0$, after executing θ at least once is:

$$x = 2(n+1) \wedge n \geq 0 \wedge 2n \leq M$$

Our exploration procedure (with acceleration) will compute this set. It will also produce a symbolic configuration graph having a loop on the set given above since this set is closed under the application of θ. However, it is clear that the iteration of θ is bounded due to the guard $x \leq M$. The problem is to infer automatically this fact and to express it as a fairness constraint on the symbolic graph. This problem can be more general than the one illustrated by this example and may concern a whole component of the symbolic graph, corresponding to nested loops and not only to a simple one. However, this problem is clearly as hard as deciding termination of programs (halting problem of Turing machines) and cannot be solved automatically in general.

The question of extending symbolic reachability techniques in order to generate finite abstractions with fairness conditions can be addressed in a similar manner for several kinds of infinite-state models. To illustrate our approach, we consider in this paper only numerical models which are *fair parametric extended automata*, i.e., automata with integer counters that can be reset, incremented, and compared against parameters. The fairness conditions are expressed on the transitions of the model under constraints on the source configurations of these transitions. Constraints on these source configurations are expressed using predicates on the variables of the system.

We show that our reachability analysis procedures and the acceleration techniques we developed in [5] can be exploited and adapted in order to deal with fairness conditions.

First, we show how the translation of fairness conditions from the original model to the abstract one can be done using symbolic reachability analysis. The idea is simple and consists in considering an extended model with additional boolean variables corresponding to the predicates involved in the fairness conditions. This forces our symbolic analysis procedure to split the abstract model according to these predicates. This makes easy the translation of the fairness conditions on the transitions of the so obtained symbolic graph, and ensures the soundness of this translation.

Then, we consider the harder problem of inferring new fairness conditions corresponding to bounded iterability of loops, and more generally to bounded iterability of some kinds of connected components (nested loops). The main intuitive idea we develop is that, to perform an acceleration step, our techniques are based on an analysis of the transitions in order to compute the effect of their iterations. From this analysis, we can actually extract informations about the bounded/unbounded iterability of these transitions. If we consider again the example mentioned above, we can determine automatically that the transition θ adds to x the value 2 each time it is executed. Then, from this fact, we first infer that, after executing $n + 1$ times the transition θ starting from $x = 0$, we get the value $x = 2(n + 1)$, and we know also that $2n < M$ should be true since the guard must be satisfied. This is basically what our acceleration techniques do in order to compute the set of reachable configurations. Now, it is also possible to infer from the fact above that θ can only be iterated a finite number of times. This can be done by deciding whether there exists an n such that the obtained value after n iterations of θ (i.e., $0 + 2n$) falsifies the guard, that is, an n such that $2n > M$. This is obviously true (in this example), which allows to infer the boundedness of the loop. We show that this basic idea allows to decide the bounded iterability of any elementary circuit in the symbolic graph. Moreover, we show that this result can be extended to some classes of nested loops (connected components), by showing that the analysis of the iterability of these components can be reduced to the analysis of simple loops.

The same approach can also be applied to timed models (with real-valued clocks) under some restrictions, essentially time-determinism (transitions in the analyzed loops must take fixed time values which may be parameterized).

We show that our techniques can be applied to many nontrivial examples. In particular, we consider a parametric timed model of the Bounded Retransmission Protocol (with unbounded counters and clocks, parametric values of the number of retransmissions, and of the timeouts), and show that our techniques generate automatically a finite abstraction of the system and fairness constraints which allow to model-check the liveness properties of the protocol, in addition to its safety properties. Other examples are considered to illustrate the case where the ability to analyze nested loops is needed.

Related work: There exist many work on the symbolic reachability analysis of infinite-state systems using acceleration techniques [8,7,9,3,2,5]. These works consider mainly the case of safety properties. In [10,18], techniques for automatic verification of liveness properties are proposed in the framework of regular model checking, for the particular case of length preserving transformations on sequences, e.g., actions of parameterized networks of processes connected sequentially. In [6], techniques for translating fairness conditions from a concrete to a finite abstract model, as well as a procedure for computing safe fairness conditions on the abstract model are proposed in the particular case of parameterized sequential networks of processes.

There exist many works on proving liveness properties, and in particular the termination of programs [14,16]. These works do not propose fully automatic techniques, but are based on a general proof principle which consists in finding a ranking function on configurations which decreases along each execution path according to a suitable well-founded ordering. Few recent works try to automatize this principle and propose heuristics for the generation of such ranking functions in some restricted cases [12,11]. These works reason about the structure of the original model, whereas our techniques are applied on the structure of the symbolic graph generated by the reachability analysis. This fact actually is important. Indeed, in our symbolic graph, according to our acceleration techniques, loops in the original model are unfolded until we find sequences of transitions that have a periodic effect. This unfolding helps for the computation of the effect of iterated transitions but also in the reasoning about the boundedness of these iterations. It turns out that our techniques are powerful enough to deal with complex cases for which, if we reason directly on the original model, it is necessary to define a ranking function which decreases according to a lexicographic ordering on the variables, which is hard to guess automatically.

Outline: In Section 2, we give some basic definitions and introduce the kind of constraints and operations used in our models. In Section 3, we introduce the model of fair parametric extended automata and its semantics. In Section 4, we recall from [5] the symbolic structures, the reachability algorithm, and the extrapolation techniques. In Section 5, we show how the concrete fairness conditions may be translated into fairness constraints at the abstract level. In Section 6, we present the conditions needed and the methods used in order to synthesize fairness conditions for simple loops of the symbolic graph. In Section 7, we show how the results for simple loops can be extended to nested loops.

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 fact 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*. Parameters can be seen as variables that are not modified by the system (they keep their initial values all the time). 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.

We consider *simple operations* on variables corresponding to sets of 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 $t \in AT(\mathcal{P})$.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be the vector of all variables under consideration. Simple operations correspond to assignements of the form $\boldsymbol{x} := A\boldsymbol{x} + \boldsymbol{b}$ where $\boldsymbol{b}$ is a n-dim vector of terms, and A is a $n \times n$ $\{0,1\}$-matrix such that each row has at most one non-zero (1) value. Let us call *simple matrices* such $n \times n$ $\{0,1\}$-matrices.

3 Fair Parametric Extended Automata

We consider in our work models which are automata supplied with integer counters and real valued clocks. Counters and clocks can be reset using simple operations, and they can tested using simple parametric constraints (they can be compared to parameters in these constraints). Moreover, we consider fairness conditions on the transitions of these models. To simplify the presentation, we focus here on the case where only counters are considered. A Parametric Extended Automaton (PEA) is a tuple $\mathcal{T} = (Q, X, P, \delta)$ where:

- Q is a finite set of *control states*,
- X is a finite set of *counters* ranging over the set positive integers $\mathbb{N}$,
- P is a finite set of *parameters* ranging over positive integers,
- δ is a finite set of *guarded actions* of the form $(q_1, g(X,P), sop, q_2)$ where $q_1, q_2 \in Q$, $g(X,P) \in SC(X,P)$ is a *guard*, and sop is a simple operation over X.

A configuration of $\mathcal{T}$ is a triplet $\gamma = \langle q, \mu, \pi \rangle$ where q is a control state, μ and π are respectively valuations of the counters and parameters. For each guarded action $\tau = (q_1, g, sop, q_2) \in \delta$ we define a transition relation between configurations: $\langle q_1, \mu_1, \pi_1 \rangle \rightarrow_\tau \langle q_2, \mu_2, \pi_2 \rangle$ iff $\pi_1 = \pi_2$, $g(\mu_1, \pi_1)$ is true, and $\mu_2 = sop(\mu_1)$. Then, a *computation sequence* is a sequence $\gamma_0 \tau_0 \gamma_1 \tau_1 \gamma_2 \ldots$ such that, for every $i \geq 0$, $\gamma_i \rightarrow_{\tau_i} \gamma_{i+1}$, and either it is infinite, or it is finite but maximal (cannot be prolonged).

We introduce *constrained fairness* conditions on PEAs expressed on their transitions. Let a *constrained action* be a pair $\langle \tau, f(X, P) \rangle$ where τ is a guarded action and $f(X, P) \in FO(X, P)$ is a formula expressing a constraint on the sources of τ. A Fair PEA is a tuple $(Q, X, P, \delta, \mathcal{W}, \mathcal{B})$, where (Q, X, P, δ) is a PEA, $\mathcal{W}$ and $\mathcal{B}$ are respectively set and set of sets of constrained actions.

The set $\mathcal{W}$ (called justice set) allows to express *weak fairness* conditions saying that whenever an action is permanently enabled, it must be executed sometime in the future (hence, it will be executed infinitely often). The set $\mathcal{B}$ (called boundedness set) allows to express *bounded iterability* conditions saying that, for each $U \in \mathcal{B}$, if *all actions* in U are taken infinitely often, then it must also be the case for some action *not* in U. Formally, using the notations $\exists^\omega$ for "there exists infinitely many", and $T(U)$ for the set of transitions appearing in U, a computation sequence $\sigma = \gamma_0 \tau_0 \gamma_1 \tau_1 \gamma_2 \ldots$ is *fair* iff:

- $\forall \langle \tau, f \rangle \in \mathcal{W}.\ \forall i \geq 0$, if $\forall j \geq i.\ \gamma_j \models f$ and $\exists \gamma.\ \gamma_j \rightarrow_\tau \gamma$ then $\exists k \geq i.\ \tau_k = \tau$,
- $\forall U \in \mathcal{B}$, if $\forall \langle \tau, f \rangle \in U.\ \exists^\omega j.\ (\tau_j = \tau$ and $\gamma_j \models f)$, then $\exists \tau' \notin T(U).\ \exists^\omega k.\ \tau_k = \tau'$.

The following example shows how constrained fairness conditions can be used to express non trivial fairness conditions.

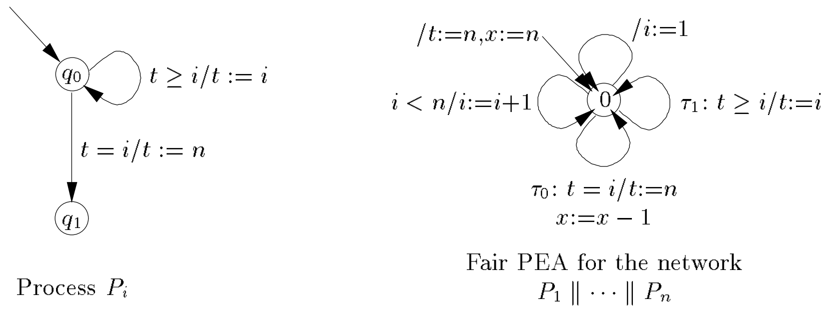


Fig. 1. Example for constrained fairness.

Example 1. Consider the parameterized network $P_1 \parallel \cdots \parallel P_n$ where each process P_i is described in Figure 1. At control state q_0, the processes are competing for getting a resource. Process P_i has the resource when $t = i$ and it moves to control state q_1 when it does not need the resource anymore. A process of smaller index can, however, preempt a process of bigger index number. It is natural to assume that a process that has the resource and is never preempted, eventually reaches control state q_1.

We are interested in verifying that eventually all processes reach q_1, under the assumption that all transitions are weak fair. The parameterized system can be modeled by a fair parametric extended automaton as shown in Figure 1. In this automaton, x counts the processes in control state q_0. The property we want then to check is whether eventually $x = 0$, i.e. $\Diamond x = 0$.

Clearly this property is not satisfied, if we do not assume any fairness on the transition τ_0 decrementing x. Assuming that this transition is weakly fair is, however, too strong because if we remove the guard $t \geq i$ in P_i, the system $P_1 \parallel \cdots \parallel P_n$ will not satisfy the property of interest but still the fair extended automaton does. In fact, what we would like to express is: if the value of t remains always unchanged, then eventually a transition decrementing x is taken. To express this property we add a variable to (for t old) to the abstract system, and add the assignment $to := t$ to the transition τ_1. Then, the constrained weak fairness condition $(\tau_0, to = t)$ expresses the property we want.

4 Symbolic Reachability Analysis

We present now the main principles of the symbolic reachability analysis techniques developed in [5]. For a PEA $\mathcal{T}$, these techniques build (when they terminate) a symbolic reachability graph $SG(\mathcal{T}) = (V, E)$ where V is a finite set of structures representing (infinite) sets of configurations of $\mathcal{T}$ and $E \subseteq V \times V$ is a finite set of transitions between structures in V, corresponding to transitions of $\mathcal{T}$. The fairness conditions are defined similarly to Fair PEA. The constrained edges of the symbolic graph are pairs $\langle e, b \rangle$ with $e \in E$ and b is a boolean expression (in $SC(\mathcal{X}, \mathcal{P})$) on the source configuration of e. A Fair $SG(\mathcal{T})$ is a tuple $(V, E, \mathcal{W}_{SG}, \mathcal{B}_{SG})$ where $\mathcal{W}_{SG}$ (the justice set) is a set of constrained edges and $\mathcal{B}_{SG}$ (the boundedness set) is a set of sets of constrained edges.

4.1 Symbolic representation structures

In order to represent sets of configurations of PEAs, we use *Constrained Parametric Difference Bound Matrices*, introduced in [5], which is an extension of the Difference Bound Matrices (DBM) [13] used for the representation of reachability sets of (non parametric) timed automata [4].

Definition 1. *Let $\mathcal{T} = (Q, X, P, \delta)$ be a PEA, let $X = \{x_1, ..., x_n\}$ be the set of counters of the PEA $\mathcal{T}$, and let x_0 be an additional counter whose value is always equal to 0. A* parametric difference bound matrix *(PDBM) which represents a set of configurations over X is a $(n + 1) \times (n + 1)$ matrix M of elements in $AT(P) \times \{<, \leq\}$. Each entry of M, $M[i, j] = (t, \prec)$ encodes the constraint $x_i - x_j \prec t$. Given a valuation π of parameters, the semantics of M is defined by $[\![M]\!]_\pi = [\![\wedge_{ij} M(i, j)]\!]_\pi$.*

A constrained PDBM*, is a pair $S = (M, \phi)$ where M is a PDBM and ϕ is a constraint in $FO(P)$. Given a valuation π of parameters, the semantics of (M, ϕ) is defined by $[\![(M, \phi)]\!] = \{(\mu, \pi) \mid \mu \in [\![M]\!]_\pi, \pi \in [\![\phi]\!]\}$.*

The extrapolation procedure [5] used to help termination of the reachability analysis introduces new parameters, called *iteration parameters*. These parameters correspond to numbers of iterations of control loops. Let N be the set of these new parameters. The definition of constrained PDBM is extended to represent sets of configurations represented by means of variables in N.

Definition 2. *An* open *PDBM is a PDBM which elements are terms in $AT(P \cup N) \times \{<, \leq\}$. An* open *constrained PDBM is a constrained PDBM (M, ϕ) where M is a open PDBM and ϕ is a constraint in $FO(P \cup N)$.*

The semantics of open PDBM and open constrained PDBM are defined similarly, by considering also valuations ν of iteration parameters in N. Given an open PDBM M (resp. constrained PDBM $S = (M, \phi)$), we denote by $Iter(M)$ (resp. $Iter(S)$) the set of iteration variables appearing in M (resp. in M or ϕ).

Definition 3. *A symbolic configuration Γ is a pair (q, S) where $q \in Q$ is a control state, and S is a constrained PDBM. A symbolic configuration $\Gamma = (q, S)$ includes $\Gamma' = (q', S')$ (notation $\Gamma \supseteq \Gamma'$) if $q = q'$ and $[\![S]\!] \supseteq [\![S']\!]$.*

[5] shows how the standard operations on DBM (transformation into a canonical form, intersection, and inclusion test) can be lifted to (open) constrained PDBM. We have implemented in C++ a package providing data structures and operations for manipulating (open) constrained PDBM over integer or real variables. This package is distributed with the tool TReX[1].

The operations on (open) constrained PDBM manipulate arithmetical terms in $AT(P)$ and boolean formulas over comparisons of the form $t \prec t'$, with $t, t' \in AT(P)$ and $\prec \in \{<, \leq\}$. We have implemented in TReX a package allowing to represent in a compact way these terms and formulas, using arithmetical and logical simplification rules (symmetry of arithmetical operations, factorization, properties of boolean connectives, etc.). Our representation also includes information about the nature of terms (real/integer, linear/non-linear) and formulas (in $LFO(X, P)$ or $RFO(X, P)$). This allows to invoke the suitable decision procedures: Omega [17] in the integer linear case, or Reduce [15] otherwise.

4.2 Building symbolic reachability graphs

Let $\mathcal{T}$ be a PEA. The procedure proposed in [5] computes, given an initial symbolic configuration Γ_0, the symbolic reachability graph $SG(\mathcal{T})$ representing the set $post^*(\Gamma_0)$. For that, starting from Γ_0, the transitions of $\mathcal{T}$ are applied, when possible, to build new vertices in $SG(\mathcal{T})$. For a transition τ of $\mathcal{T}$ and a symbolic configuration Γ, we denote by $post_\tau(\Gamma)$ the vertex computed by applying τ to Γ. The new vertices of $SG(\mathcal{T})$ are treated according to a depth-first traversal. The construction stops when each new generated vertex is covered by (included in) some already computed vertex.

During this construction, an *extrapolation* technique is used to help termination. The extrapolation technique defined in [5] 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 non reachable configurations). We present below the main idea of the extrapolation technique.

[1] `http://www.liafa.jussieu.fr/~sighirea/trex` and `http://www-verimag.imag.fr/~annichin/trex`

Let θ be a control loop in $\mathcal{T}$, i.e., a path $\tau_0 \dots \tau_k$ with $\forall i \in \{0, \dots, k\}, \tau_i = (q_i, g_i, sop_i, q'_i)$, such that $q = q_0 = q'_k$ and $\forall i \in \{0, \dots, k-1\}$, $q'_i = q_{i+1}$. Let $\Gamma = (q, S = (M, \phi))$ be a symbolic configuration. We note by $post_\theta(\Gamma)$ the computation $(post_{\tau_k} \circ \dots \circ post_{\tau_0})(\Gamma)$. Suppose that 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 $\Delta' = M_2 - M_1$. If for all values of parameters $\Delta = \Delta'$, we may guess that the symbolic configuration obtained after n iterations of θ is $post^n_\theta(q, S) = (q, S_n = (M + n * \Delta, \phi \wedge n \geq 0))$. To check the exactness of this guess, we have to check that for all $n \geq 0$, the application of θ on (q, S_n) has an effect of adding Δ, provided the guards in θ are satisfied. This corresponds to the induction step for the proof of exactness. If the extrapolation can be applied, $post^n_\theta(q, S_n)$ is added to the set of vertices V in $SG(\mathcal{T})$, and the transition $((q, S_1), post^n_\theta(q, S_n))$ to the set of edges E.

5 Translation of Fairness Constraints

We explain how one can adapt the symbolic reachability analysis presented in Section 4 to take into account concrete fairness conditions. To do so, consider a set of constrained actions $\langle \tau_i, f_i \rangle$, for $i = 1, \dots, n$. We associate to each predicate f_i a boolean variable b_i. Then, we extend the symbolic configurations Γ to $\langle \Gamma, \nu \rangle$, where ν is a valuation of the b_i's such that if $\nu(b_i)$ is true (*resp.* false), then there is concrete configuration in Γ that satisfies (*resp.* falsifies) f_i. Notice that the same symbolic configuration Γ may be extended to many extended configurations. Now, we have to lift the predicate transformers $post_\tau$ and $post_\theta$ to extended configurations. Let us first deal with the simple case of $post_\tau$. We then define $post^f_\tau$ as the function that takes an extended configuration $\langle \Gamma, \nu \rangle$ as argument and yields a set $post^f_\tau(\langle \Gamma, \nu \rangle)$ of extended configurations as result. The set $post^f_\tau(\langle \Gamma, \nu \rangle)$ is such that the following conditions are satisfied:

1. If $\langle \Gamma', \nu' \rangle \in post^f_\tau(\langle \Gamma, \nu \rangle)$, then $post_\tau(\Gamma) = \Gamma'$,
2. if $\langle \Gamma', \nu' \rangle \notin post^f_\tau(\langle \Gamma, \nu \rangle)$, then there is no configuration in Γ that evaluates $(f_1, \cdots, f_n)$ to ν'.

Now, the symbolic reachability graph contains an edge labeled by τ from $\langle \Gamma, \nu \rangle$ to all extended configurations in $post^f_\tau(\langle \Gamma, \nu \rangle)$. The definition of $post^f_\theta$ is similar except that in the symbolic reachability graph we will also have edges connecting all configurations in $post^f_\theta(\langle \Gamma, \nu \rangle)$.

Now, we can explain how to translate the concrete fairness conditions to fairness conditions on the symbolic reachability graph. Thus, assume that $\langle \tau, f \rangle$ is in the set $\mathcal{W}$ of constrained weak fairness conditions. Then, when verifying a property on the symbolic reachability graph, we only need to consider paths that contain infinitely many τ-edges, if from some point on they only contain extended configurations where the boolean associated to f evaluates to *true*. In other words, we will have that $\mathcal{W}_{SG}$ contains the constrained edge $\langle e_\tau, b_f \rangle$, where e_τ stands for the edges labeled by τ and b_f is the boolean associated to f. Similarly, for a set $U = \{\langle \tau_i, f_i \rangle \mid 1 \leq i \leq n\} \in \mathcal{B}$ expressing a boundedness condition, we will have that $\{\langle e_{\tau_i}, \vee^n_{i=1} b_i \rangle \mid 1 \leq i \leq n\} \in \mathcal{B}_{SG}$.

6 Synthesis of Fairness Constraints for Elementary Loops

We explain how boundedness conditions can be synthesized for the loops of the symbolic graph which do not correspond to any legal infinite computation in the original fair PEA.

Let consider the simplest case of a self-loop θ (i.e., a loop with only one transitions). To show that θ is finitely (*boundedly*) iterable, the idea is to show that for all configurations γ_0 belonging to the initial symbolic configuration, all executions paths resulting from iterating θ stop. Now, suppose that given an arbitrary number $n \geq 0$, we can compute the configuration $\theta^n(\gamma_0)$ obtained after n iterations of θ starting from γ_0 Then, deciding that θ is boundedly iterable can be reduced to decide if for all initial configurations γ_0, there exists an $n \geq 0$ such that the configuration $\theta^n(\gamma_0)$ does not satisfy the guard of θ.

We will show that, under some conditions, the analysis of complex connected components of the symbolic graph may be reduced to the above idea for analysis of self-loops. For this, we proceed by induction on the complexity of connected component to be analyzed, C. The simpler case are elementary circuits, for which we define the *degree of complexity*, $degree(C) = 1$. For a non-elementary connected component, its degree is defined inductively by $degree(C) = max_{C' \in C}(degree(C')) + 1$ for all C' connected (strict) subcomponent of C. In this section we consider elementary circuits.

6.1 Computation of the effect

Let $\theta = \Gamma_0 \tau_0 \Gamma_1 \tau_1 \Gamma_2 ... \tau_{k-1} \Gamma_k$ be a computation sequence of the symbolic graph such that θ is an elementary circuit, i.e., $\Gamma_0 \supseteq \Gamma_k$ and $\forall 0 \leq i < j < k.\ \ \Gamma_i \not\supseteq \Gamma_j$.

Let $\tau = (q_1, g(X, P), sop, q_2)$ be a transition, and let *sop* correspond to the assignement $\boldsymbol{x} := A_\tau \boldsymbol{x} + \boldsymbol{b_\tau}$ of the counters (including x_0) where A_τ is $(n+1) \times (n+1)$ simple matrix, and $\boldsymbol{b_\tau}$ is a $(n+1)$-vector of terms in $AT(\mathcal{P})$.

We can lift operations on vectors of counters to operations on PDBMs in the following manner. First, we transform the vector $\boldsymbol{b_\tau}$ into a $(n+1) \times (n+1)$ matrix of terms B_τ defined by: $\forall 0 \leq i, j \leq n.\ \ B_\tau[i,j] = \boldsymbol{b_\tau}(i) - \boldsymbol{b_\tau}(j)$. Then, applying the assignement $\boldsymbol{x} := A_\tau \boldsymbol{x} + \boldsymbol{b_\tau}$ on a set of configurations described by a PDBM M leads to a PDBM $M' = A_\tau * M * A_\tau^T + B_\tau$ where A_τ^T is the transposed matrix of A_τ.

In this way, the effect of a transition of the symbolic graph on a PDBM is given by a pair of matrices $(A, B) \in \mathsf{A} \times \mathsf{B}$, where A is the class of simple matrices, and B is defined by:

$$\begin{aligned} \mathsf{B} = \{B \mid \forall 0 \leq i, j \leq n.\ \ & B[i,j] \in AT(P \cup N) \ \wedge\ B[i,j] = -B[j,i] \ \wedge \\ & \forall 0 \leq k \leq n.\ \ B[i,k] + B[k,j] = B[i,j]\} \end{aligned}$$

Matrices in A and B have nice properties. Indeed, A is closed under multiplication of matrices, and B is closed under addition and closed under multiplication with matrices in A. Notice that, matrices in B represent parameterized configurations of counters (they encode single vectors of terms). We define the mapping $cf : \mathsf{B} \rightarrow (AT(P \cup N))^{n+1}$, which maps each matrix in B to the corresponding parameterized configuration for counters.

Using these properties of sets A and B, it is easy to show that the effect of a sequence of transitions $\tau_0\tau_1...$ on a PDBM is also given by a pair of matrices $(A, B) \in \mathsf{A} \times \mathsf{B}$. Consequently, starting with a PDBM $M_0 \in \Gamma_0$ and iterating one time θ gives a new PDBM $\theta(M_0) = A_\theta * M_0 * A_\theta^T + B_\theta$ where the pair $(A_\theta, B_\theta) \in \mathsf{A} \times \mathsf{B}$ cumulates the effect of the sequence of transitions composing θ. The following proposition states the conditions under which $\theta^n(M_0)$ can be computed.

Proposition 1. *Let $\theta = \Gamma_0\tau_0\Gamma_1\tau_1\Gamma_2...\tau_{k-1}\Gamma_k$ be an elementary circuit in a symbolic graph such that $\Gamma_0 = (q_0, (M_0, \phi_0))$ and the effect of the sequence of transitions $\tau_0, ..., \tau_{k-1}$ is given by matrices (A_θ, B_θ). If the following PDBM $\Delta_\theta = A_\theta * M_0 * A_\theta^T + B_\theta - M_0$ satisfies the conditions:*

- *SC_1: $A_\theta * \Delta_\theta * A_\theta^T = \Delta_\theta$ and*
- *SC_2: Δ_θ is a closed parameterized configuration for counters, i.e., $\Delta_\theta \in \mathsf{B}$ and is closed (i.e., does not contain variables in N)*

then, $\forall \boldsymbol{x_0} \in [\![M_0, \phi_0]\!]\,.\ \forall n \geq 0.\ \ \theta^n(\boldsymbol{x_0}) = \boldsymbol{x_0} + n * cf(\Delta_\theta)$.

Both conditions SC_1 and SC_2 correspond to conjunctions of equalities between arithmetical expressions over integers. These formulas are decidable in the linear case, i.e., for LFO formulas. Condition SC_1 is similar to the first condition for exactness of extrapolation (see Section 4). Condition SC_2 means that the effect of each iteration is *deterministic*, that is, each iteration of θ has the effect of adding the same (parametric) value defined by Δ_θ.

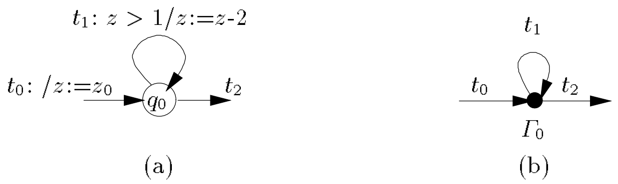


Fig. 2. A simple parameterized extended automaton (a) and its symbolic graph (b).

Example 2. We apply Proposition 1 for the extended automaton on Figure 2 (a), which symbolic graph contains only one self-loop $\theta = \Gamma_0\ t_1\ \Gamma_0$. For this circuit $A_\theta = I$ and $B_\theta = \begin{pmatrix} 0 & 2 \\ -2 & 0 \end{pmatrix} \in \mathsf{B}$. For all M_0, $\Delta_\theta = A_\theta * M_0 * A_\theta^T + B_\theta - M_0 = B_\theta$ which satisfies trivially the conditions SC_1 and SC_2. By Proposition 1 we have $\theta^n(\boldsymbol{x_0}) = \boldsymbol{x_0} + n * (0, -2)$, since $cf(\Delta_\theta) = (0, -2)$.

Remark 1. The computation presented above can be extended to the case of Parameterized Timed Automata for time-deterministic circuits. Since the evolution of clock variables is given not only by the operations on transitions, but also by the guards and the invariants, the computation of the effect of loops should take into account these elements. However, the computation of Δ_θ is already done during the symbolic reachability algorithm for extrapolation. We have to test in addition that Δ_θ satisfies condition SC_2.

6.2 Synthesis of fairness constraints

We say that an elementary circuit θ is *boundedly iterable* in the configuration Γ_0 if there is an integer $n \geq 0$ such that $\theta^n(\Gamma_0) = \emptyset$.

By Proposition 1, we know that for every parametric configuration $\boldsymbol{x_0} \in \Gamma_0$, $\theta^n(\boldsymbol{x_0}) = \boldsymbol{x_0} + n * cf(\Delta_\theta)$. Let us denote by $\tau_{[0,i-1]}$ the sequence of transitions $\tau_0 \ldots \tau_{i-1}$, for any $i \in \{0, \ldots, k-1\}$, and by $\tau_{[0,i-1]}(\boldsymbol{x})$ the effect of the sequence of transitions $\tau_0 \ldots \tau_{i-1}$ on $\boldsymbol{x}$ (if $i = 0$, we consider that no transition is applied). Then, we have:

Proposition 2. *Let θ be an elementary circuit such that Δ_θ satisfies SC_1 and SC_2. Then θ is boundedly iterable iff $\forall \boldsymbol{x_0} \in \Gamma_0.\ \exists n \geq 0.\ \exists i \in \{0, \ldots, k-1\}.\ \tau_{[0,i-1]}(\boldsymbol{x_0} + n * cf(\Delta_\theta)) \not\models guard(\tau_i)$.*

The formula given by Proposition 2 have an equivalent form in terms of operations on PDBM:

$$\begin{array}{ll} \forall P \forall N \forall X_0. & \phi_0(P, N) \wedge M_0(P, N, X_0) \implies \\ & \exists n \geq 0.\ \ \bigvee_{i=0}^{k-1} \tau_{[0,i-1]}(M_0 + n * \Delta_\theta) \cap guard(\tau_i) = \emptyset \end{array}$$

The validity of this formula is decidable in the linear case, since all variables are integers. If the formula above is valid, we synthesize a boundedness constraint $T(\theta) \in \mathcal{B}_{SG}$ for the set $T(\theta) = \{\tau_0, \ldots, \tau_{k-1}\}$ of transitions in θ.

From the test given in Proposition 2, we can build a formula $\phi_i^{\theta,\Gamma_0}(\boldsymbol{x}, n)$ giving the condition under which after iterating n times θ on the configuration $\boldsymbol{x} \in \Gamma_0$, the execution stops at the transition τ_i:

$$\phi_i^{\theta,\Gamma_0}(\boldsymbol{x_0}, n) = \tau_{[0,i-1]}(\boldsymbol{x_0} + n * cf(\Delta_\theta)) \not\models guard(\tau_i)$$

which is easy to translate in terms of operations on PDBM.

Example 3. We consider here an example which is inspired from the model of the Bounded Retransmission Protocol. It is a system with two clocks c_1 and c_2 and a counter x (see Figure 3 (a)). Intuitively, c_1 represents the clock of a

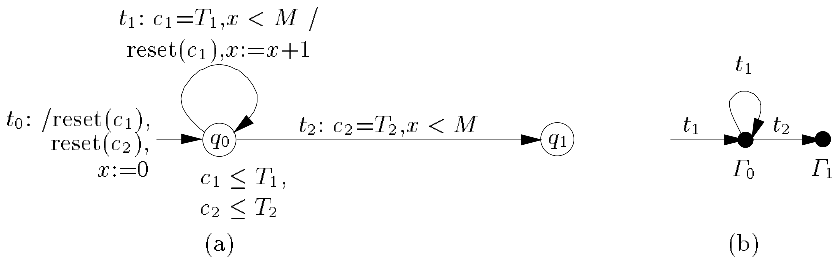


Fig. 3. An elementary circuit of BRP.

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 $q_0 t_1 q_0$ is performed. The transition t_1 corresponds in BRP to a retransmission action by the sender. The number of these retransmissions is bounded by M which is an integer parameter. The question is whether the state q_1 (considered as a bad state) is reachable. Roughly speaking, this corresponds to a property of synchronization between the sender and the receiver in BRP: the timeout of the receiver should not expire before the sender has finished all its retransmissions.

The symbolic graph generated for this example is given on Figure 3 (b), where $\Gamma_0 = (q_0, S_0)$ and $\Gamma_1 = (q_1, S_1)$ with

$$\begin{aligned} S_0 \equiv (\ M_0 = \ & (0 \leq c_1 \leq T_1 \wedge (n+1) * T_1 \leq c_2 \leq (n+2) * T_1 \wedge, \\ & c_1 - c_2 = (n+1) * T_1 \wedge x = n+1) \\ \phi_0 = \ & (0 \leq n \wedge 0 \leq T_1 \wedge (n+2) * T_1 \leq T_2 \wedge n < M)\ ) \\ S_1 \equiv (\ M_1 = \ & (0 \leq c_1 \leq T_1 \wedge T_2 \leq c_2 \wedge c_1 - c_2 = (n+1) * T_1 \wedge x = n+1), \\ \phi_1 = \ & (0 \leq n \wedge 0 \leq T_1 \wedge (n+1) * T_1 \leq T_2 \leq (n+2) * T_1 \wedge n+1 < M)\ ) \end{aligned}$$

The presence of the unfair loop in Γ_0, $\theta = \Gamma_0 t_1 \Gamma_0$, fails the verification of the reachability property for q_1. By analyzing θ, we find that Δ_θ (on both clocks and counters) is a parameterized configuration and $\Delta_\theta(c_1) = 0$, $\Delta_\theta(c_2) = T_1$, $\Delta_\theta(c_2 - c_1) = T_1$, and $\Delta_\theta(x) = 1$. By Proposition 2 we have that θ is boundedly iterable iff the formula:

$$\forall T_1, T_2, c_1, c_2.\ \forall M, x, n.\ \ \phi_0 \wedge M_0 \implies \exists n' \geq 0.\ \ x + n' \geq M$$

After the elimination of real variables (T_1, T_2, c_1, c_2), we obtain the following linear formula on integer variables:

$$\forall M, x, n.\ \ 0 \leq n \wedge n < M \wedge x = n+1 \implies \exists n' \geq 0.\ \ x + n' \geq M$$

which is clearly valid. Hence, we can generate a boundedness condition for this loop, saying that if t_1 is taken infinitely often, then transitions different from t_1, including t_2, are also taken infinitely often.

7 Synthesis of Fairness Constraints for Nested Loops

Fairness constraints generation based on analysis of elementary circuits is sufficient for dealing with a wide class of systems (e.g., BRP). However, there are cases which require reasoning about arbitrary combination of elementary circuits, i.e., connected components.

Consider, for example, the program of Figure 4 and its PEA model. The approaches [12, 11] based on ranking functions will fail to prove the termination of this program. Indeed, the ranking functions synthesized for the innermost and the outermost **while** statements cannot be ordered according to a lexicographic order using the structure of the program: the behaviour of the inner depends on the value of y computed in the outermost loop, which behaviour depends of the value of z computed in the innermost loop. In our approach, the reachability algorithm applied on the extended automaton of Figure 4 generates the symbolic graph given on Figure 5 and Table 1. In this graph, we represent by dotted transitions a sequence of transitions. For example, configuration Γ_1 is reached from Γ_0 by executing the sequence t_0, t_1. The symbolic graph has several self-loops

```
x:=X0; y:=Y0; z:=0;
while x > 0 do
   z:=y;
   while z > 1 do z:=z-2; od;
   if z = 0 then
      x:=x-1; y:=y+3;
   else
      y:=y-1; z:=0;
   fi;
od;
```

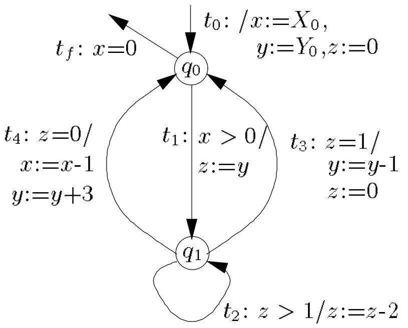


Fig. 4. Example of nested loops.

build from transition t_2. Using the method presented in Section 6, we can infer that these loops are boundedly iterable, which means that the innermost **while** statement always stops. The connected components starting in Γ_0' and Γ_0'' correspond to the outermost **while** statement for respectively odd and even values of the parameter Y_0. Indeed, from the initial configuration Γ_0, we unfold one time the outermost **while** statement. After executing the sequence $t_0t_1t_2t_2^*$ (the parameter n_0 is introduced by iterating t_2 in Γ_2), we have two cases depending on which transition (t_3 or t_4) is chosen in configuration Γ_2. If transition t_3 is chosen, we deduce that Y_0 is odd (see constraint ϕ_0' in Table 1). By unfolding once more the outermost **while** statement (i.e., the sequence $t_1t_2t_2^*t_4$), we generate a configuration which differs from Γ_0 by the vector $(x_0 = 0, x = -1, y = 2, z = 0)$. After two more unfoldings, the outermost **while** produces the same effect. The extrapolation technique generates the configuration Γ_0' and the connected component $t_1t_2t_2^*t_3t_1t_2t_2^*t_4$ starting in Γ_0'. The effect of each of the components

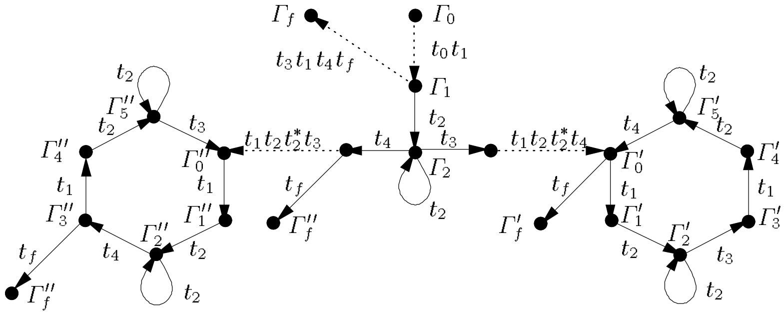


Fig. 5. Symbolic graph for the example of nested loops.

starting in Γ_0' and Γ_0'' is to add the vector $cf(\Delta) = (0, -1, 2, 0)$, which satisfies the conditions SC_1 and SC_2 of Proposition 1. This is due to the fact that the effect of innermost loops is reset by the transitions t_3 and t_4. However, these components are not elementary circuits. Then, we need to generalize the techniques given in Section 6 in order to deal with a whole connected component. We show

Table 1. Main symbolic configurations for graph on Figure 5.

conf.	*ctrl. state*	*PDBM* x	y	z	*constraint*
Γ_0	q_0	X_0	Y_0	0	$\phi_0 \equiv X_0, Y_0 \geq 0$
Γ'_0	q_0	X_0-n'_1-1	Y_0+2n'_1+2	0	$\phi'_0 \equiv Y_0 = 2n_0{+}3, X_0, Y_0, n'_1, n_0 \geq 0$
Γ'_1	q_1	X_0-n'_1-1	Y_0+2n'_1+2	Y_0+2n'_1+2	ϕ'_0
Γ'_2	q_1	X_0-n'_1-1	Y_0+2n'_1+2	Y_0+2n'_1-2n'_2	ϕ'_0,$n'_2 \geq 0$
Γ'_3	q_0	X_0-n'_1-1	Y_0+2n'_1+1	0	ϕ'_0
Γ'_4	q_1	X_0-n'_1-1	Y_0+2n'_1+1	Y_0+2n'_1+1	ϕ'_0
Γ'_5	q_1	X_0-n'_1-1	Y_0+2n'_1+1	Y_0+2n'_1-2n'_3-1	ϕ'_0,$n'_3 \geq 0$
Γ''_0	q_0	X_0-n''_1-1	Y_0+2n''_1+2	0	$\phi''_0 \equiv Y_0 = 2n_0, X_0, Y_0, n''_1, n_0 \geq 0$
Γ''_1	q_1	X_0-n''_1-1	Y_0+2n''_1+2	Y_0+2n'_1+2	ϕ''_0
Γ''_2	q_1	X_0-n''_1-1	Y_0+2n''_1+2	Y_0+2n''_1-2n''_2	ϕ''_0,$n''_2 \geq 0$
Γ''_3	q_0	X_0-n''_1-2	Y_0+2n''_1+5	0	ϕ''_0
Γ''_4	q_1	X_0-n''_1-2	Y_0+2n''_1+5	Y_0+2n''_1+5	ϕ''_0
Γ''_5	q_1	X_0-n''_1-2	Y_0+2n''_1+5	Y_0+2n''_1-2n''_3+ 3	ϕ''_0,$n''_3 \geq 0$

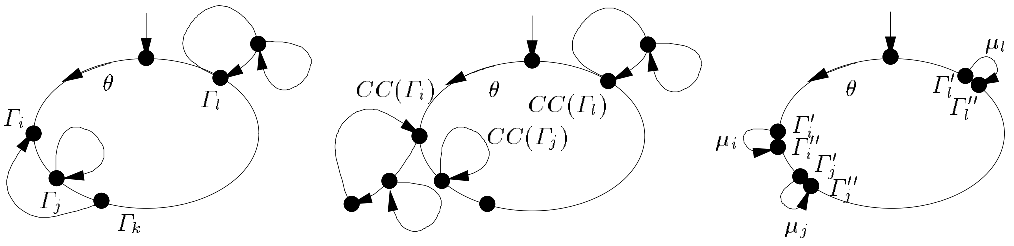
hereafter that, under some conditions, reasoning about connected components can be reduced to reasoning about elementary circuits.

7.1 Computation of the effect

Case 1. Consider the (simpler) case of a connected component C having a unique elementary circuit θ containing the initial configuration of C, $\Gamma_0 = (q_0, (M_0, \phi_0))$. The connected subcomponents of C different from θ may share zero or more symbolic configurations with θ (see first part of Figure 6).

We build from C a connected component C_θ having the same degree and execution paths that C, but where the connected subcomponents different from θ share with θ at most one symbolic configuration. For this, let $\theta = \Gamma_0\tau_0\Gamma_1\tau_1\Gamma_2...\tau_{k-1}\Gamma_0$. We order the configurations of θ according to the transition relation, i.e., $\Gamma_0 \prec \Gamma_1 \prec \Gamma_2... \prec \Gamma_{k-1}$. Since θ is the unique elementary circuit in C containing Γ_0, the unique path relating Γ_i to Γ_j with $i < j$ is contained in θ. Let Γ_i be a configuration of θ shared with a subcomponent of C. We note by $CC(\Gamma_i)$ the maximal connected subcomponent of C not including θ, such that Γ_i is the minimal (in sense of $\prec$) configuration of $CC(\Gamma_i)$ shared with θ. Then, C_θ is build from θ and, for each configuration Γ_i of θ, we put a copy of $CC(\Gamma_i)$ such that the only configuration shared with θ is Γ_i.

By construction, it follows that execution paths of C and C_θ are equal. This means that if C_θ is boundedly iterable, C is also boundedly iterable. The result of iterating one time C on a configuration $\boldsymbol{x}$ is defined by $C(\boldsymbol{x}) = \{\boldsymbol{x'} \mid \exists n_1...n_{k-1} \geq 0.\ \ \boldsymbol{x'} = (\tau_{k-1} \circ CC(\Gamma_{k_1})^{n_{k-1}} \circ ... \circ CC(\Gamma_1)^{n_1} \circ \tau_0)(\boldsymbol{x})\}$. Clearly, this set has cardinality ≥ 1, since the iteration parameters introduce non-deterministic behaviours. In order to be able to compute this set, we need to introduce additional conditions ensuring that applying C gives always the same result, i.e., its effect is deterministic.



C $\quad$ C_θ $\quad$ $\hat{\theta} = \theta$ + meta-transitions

Fig. 6. Building $\hat{\theta}$ for C.

Suppose that it is possible to compute for each subcomponent $CC(\Gamma_i)$ of C_θ the effect $\Delta_{CC(\Gamma_i)}$ satisfying the conditions of Proposition 1. Then, we can replace the subcomponent $CC(\Gamma_i)$ by a meta-transition μ_i, which operation is $\boldsymbol{x'} := \boldsymbol{x} + n * cf(\Delta_{CC(\Gamma_i)})$ with $n \geq 0$ (a new, unused $n \in N$ for each meta-transition). The meta-transition μ_i is introduced by splitting Γ_i in two: Γ_i' and Γ_i'', the source resp. the target of μ_i.

In this way, we obtain an elementary circuit, called $\hat{\theta}$, which is built with transitions from θ and with meta-transitions involving iteration parameters $n \in \boldsymbol{N}$. The result of Proposition 1 can be applied to this circuit and we obtain that:

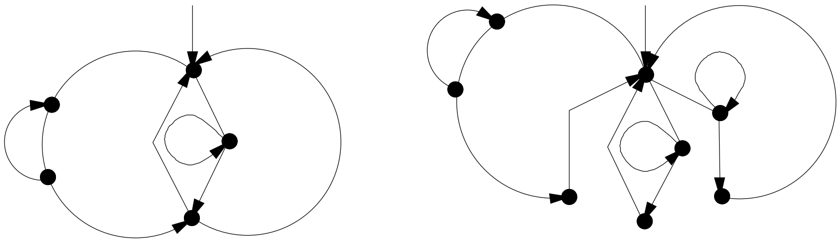
Proposition 3. *If $\Delta_{\hat{\theta}} = A_{\hat{\theta}} * M_0 * A_{\hat{\theta}}^T + B_{\hat{\theta}} - M_0$ satisfies conditions SC_1 and SC_2, then $\forall \boldsymbol{x_0} \in [\![M_0, \phi_0]\!]\, \forall n \geq 0.\;\; C^n(\boldsymbol{x_0}) = \boldsymbol{x_0} + n * cf(\Delta_{\hat{\theta}})$.*

We recall that condition SC_2 asks that $\Delta_{\hat{\theta}}$ is a configuration and it does not contain iteration parameters, so the set $C(\boldsymbol{x})$ has cardinality 1. This is possible if the circuit θ contains transitions with reset operations such that the effect of meta-transitions is canceled.

Example 4. Consider the connected component C' containing Γ_0' in the symbolic graph given on Figure 4 (b). It corresponds to an odd initial value for Y_0 and has only one elementary circuit θ' containing Γ_0'. The elementary loops in Γ_2' and Γ_5' can be reduced to meta-transitions. Moreover, the effect computed using $\hat{\theta}'$ is a configuration with $\Delta_{C'}(x) = -1$, $\Delta_{C'}(y) = 2$, and $\Delta_{C'}(z) = 0$. By Proposition 3, we have that $\forall \boldsymbol{x_0} \in \Gamma_0 \forall n \geq 0.\;\; C'^n(\boldsymbol{x_0}) = \boldsymbol{x_0} + n * cf(\Delta_{C'})$. The same computation may be done for connected component corresponding to even values of Y_0.

Remark 2. In some cases, we can deal with the presence of iteration parameters in $\Delta_{\hat{\theta}}$. The simpler case is when all operations in C cumulate values in counters, i.e., all assignments have the form $x := x + t$ (there are no reset operations). Indeed, in absence of reset operations, the (meta)-transitions in $\hat{\theta}$ cumulate their results, and the effect of the circuit is given by: $C^n(\boldsymbol{x}) = \boldsymbol{x} + \Sigma_{1 \leq i < \infty} n_i * cf(\Delta_i)$, where $\Delta_i \in \mathsf{B}$ are build from terms of assignments in (meta-)transitions.

Case 2. Consider a connected component C having several elementary circuits including its initial configuration $\Gamma_0 = (q_0, (M_0, \phi_0))$. Let $\Theta(C)$ be the set of these elementary circuits. In order to analyze C, we build from C a connected component $\tilde{C}$ having the same degree and execution paths that C, but where each connected subcomponent including Γ_0 contains only one elementary circuit of $\Theta(C)$. Moreover, each pair of such subcomponents shares only Γ_0 (see Figure 7). For this, let θ_i be an elementary circuit in $\Theta(C)$. With θ_i, we consider all maximal connected subcomponents sharing states with θ_i and without another elementary circuit in $\Theta(C)$. This will form the maximal connected subcomponent of C built around θ_i. We note by $CC(\theta_i)$ this subcomponent and by $EC(C)$ the set of all these subcomponents, $EC(C) = \{CC(\theta_i) \mid \theta_i \in \Theta(C)\}$. Then, $\tilde{C}$ is build from $EC(C)$ by splitting states such that the only state shared by each pair of subcomponents $CC(\theta_i)$ is the initial configuration Γ_0.



Connected component C Connected component $\tilde{C}$

Fig. 7. Building $\tilde{C}$.

It is easy to show that C and $\tilde{C}$ have the same execution paths. This means that if $\tilde{C}$ is boundedly iterable, C is also boundedly iterable. The paths build by iterating one time C on $\boldsymbol{x}$ are given by $C(\boldsymbol{x}) = \cup_{\theta_i \in \Theta(C)} (\cup_{j \neq i} CC(\theta_j))^* CC(\theta_i)$.

Now, if we consider separately each subcomponent $CC(\theta_i)$ of $\tilde{C}$, we can use Proposition 3 to compute its effect, and to reduce it to a meta-transition. In the best case, where all the subcomponents $CC(\theta_i)$ may be reduced to a meta-transition, we can compute, for each $\theta_i \in \Theta(C)$ the effect of $(\cup_{j \neq i} CC(\theta_j))^* CC(\theta_i)$, noted $\Delta_{CC^+(\theta_i)}$. Then, iterating C an arbitrary number of times n increments $\boldsymbol{x}$ by a parameterized configuration chosen non-deterministically in the set $\{n_1 * \Delta_{CC^+(\theta_1)}, \ldots, n_l * \Delta_{CC^+(\theta_l)} \mid n_1 \geq 0, \ldots, n_l \geq 0\}$. To be able to characterize C^n, we require that all these values be equal.

Theorem 1. *Let C be a connected component with degree$(C) \geq 1$, and $\Gamma_0 = (q_0, (M_0, \phi_0))$ its initial configuration. If C satisfies the following two conditions:*

- *Each strict subcomponent of C can be reduced to a meta-transitions.*
- *For all elementary circuit θ_j containing the initial state of C, the effect $\Delta_{CC(\theta_j)}$ satisfies conditions SC_1 and SC_2 and are all equal to a closed parameterized configuration Δ_C.*

Then, $\forall \boldsymbol{x_0} \in [\![M_0, \phi_0]\!]\, \forall n \geq 0.\ \ C^n(\boldsymbol{x_0}) = \boldsymbol{x_0} + n * cf(\Delta_C)$.

Proof. If it exists only one elementary circuit including Γ_0, the result follows from Proposition 3. If C contains more than one elementary circuits including Γ_0, we compute its effect using $\tilde{C}$ build like above. The execution of $\tilde{C}$ is given by $\Sigma_i(\cup_{j\neq i}CC(\theta_j))^*CC(\theta_i)$, i.e., the sum of executions including at least an elementary circuit. By hypothesis, we can compute $\Delta_{CC(\theta_j)}$. Moreover, we have that all $\Delta_{CC^+(\theta_j)}$ satisfies SC_1, SC_2, and are equal to Δ_C. Then, using the definition of the execution of $\tilde{C}$, it follows that for any number of iterations n of $\tilde{C}$, iterations executed in any order, $\tilde{C}^n(\Gamma_0) = \Gamma_0 + n * \Delta_C$. Since C and $\tilde{C}$ have the same execution paths and Δ_C represents a parameterized configuration (and not a zone), the result follows.

7.2 Synthesis of fairness constraints

We say that a connected component C is *boundedly iterable* in the configuration Γ_0 if there is an integer $n \geq 0$ such that $C^n(\Gamma_0) = \emptyset$.

We show now how fairness constraints can be synthesized if C satisfies the hypothesis of Theorem 1. The idea is to synthesize a formula for C expressing the fact that all execution paths of C stop. By Theorem 1, we know that for every parametric configuration $\boldsymbol{x_0} \in \Gamma_0$, $C^n(\boldsymbol{x_0}) = \boldsymbol{x_0} + n * cf(\Delta_C)$. Moreover, we can built from C the component $\tilde{C}$ (see 7.1), which has the same execution paths like C and disjoint connected subcomponents in $EC(C)$. For each connected subcomponent of $EC(C)$, let $\theta = \Gamma_0\tau_0\Gamma_1...\Gamma_{k-1}\tau_{k-1}\Gamma_0$ (in $\Theta(C)$) be its elementary circuit including the initial configuration of C. Following the method described in Section 6, we build from θ and the meta-transitions μ_i ($0 \leq i < k$) corresponding to the connected component $CC(\Gamma_j)$, the elementary circuit $\hat{\theta} = \Gamma'_0\mu_0\Gamma''_0\tau_0\Gamma'_1...\Gamma'_{k-1}\mu_{k-1}\Gamma''_{k-1}\tau_{k-1}\Gamma'_0$. We denote by $\hat{\theta}_{[0,j-1]}$ the sequence $\mu_0\tau_0...\mu_{j-1}\tau_{j-1}\mu_j$. Since all subcomponents of C may be reduced to meta-transitions, we can construct for each subcomponent $CC(\Gamma_j)$ the formula $\phi_l^{CC(\Gamma_j),\Gamma_j}(\boldsymbol{x}, n)$ which gives the condition under which after iterating n times $CC(\Gamma_j)$ on the configuration $\boldsymbol{x} \in \Gamma_j$, the execution of $CC(\Gamma_j)$ stops at its l-th configuration.

We show that we can construct from these formulas another formula $\phi_j^{\hat{\theta},\Gamma_0}(\boldsymbol{x}, n)$ saying that after iterating n times $\hat{\theta}$ on the configuration $\boldsymbol{x} \in \Gamma_0$, the execution of $\hat{\theta}$ stops at its j-th configuration ($0 \leq j \leq k-1$). This formula is the disjunction of three formula expressing all the possible cases in which the execution may stop in configuration Γ_j.

Case 1. None of edges outgoing from Γ_j and remaining in $CC(\theta_i)$ can be taken:

$$\phi_{j,(1)}^{\hat{\theta},\Gamma_0}(\boldsymbol{x}, n) = \exists m_0, ..., m_j \geq 0. \bigwedge_{\tau:\Gamma_j \to_\tau \Gamma, \Gamma \in CC(\theta)} \hat{\theta}_{[0,j-1]}(\boldsymbol{x} + n * cf(\Delta_C)) \not\models guard(\tau)$$

The integer variables $m_0, ..., m_j$ are iteration parameters used in the meta-transitions of the sequence $\hat{\theta}_{[0,j-1]}$. We can quantify existentially all these variables due to the determinism of the symbolic graph.

Case 2. The execution stops into the connected subcomponent $CC(\Gamma_j)$, but not in the first configuration of this component, i.e., Γ_j:

$$\phi_{j,(2)}^{\hat{\theta},\Gamma_0}(\boldsymbol{x}, n) = \exists m_0, ..., m_j, m \geq 0. \bigvee_{l\neq 0} \phi_l^{CC(\Gamma_j),\Gamma_j}(\hat{\theta}_{[0,j-1]}(\boldsymbol{x} + n * cf(\Delta_C)), m)$$

Case 3. The execution of $CC(\Gamma_j)$ does not terminate, which means that the hypothesis of infinitely often execution of *all* transitions in C is not satisfied:

$$\phi_{j,(3)}^{\hat{\theta},\Gamma_0}(\boldsymbol{x},n) = \exists m_0,...,m_j \geq 0.\ \forall m \geq 0.\ \neg\phi^{CC(\Gamma_j),\Gamma_j}(\hat{\theta}_{[0,j-1]}(\boldsymbol{x}+n*cf(\Delta_C)),m)$$

Proposition 4. *Let $CC(\theta)$ be a connected component having a unique elementary circuit θ such that we can compute its effect $\Delta_{CC(\theta)}$. Then $CC(\theta)$ is boundedly iterable iff $\forall P \forall N \forall \boldsymbol{x_0}.\ \Gamma_0(\boldsymbol{x_0},P,N) \Longrightarrow \exists n \geq 0.\ \vee_{j\in\{0,\ldots,k-1\}}\phi_j^{\hat{\theta},\Gamma_0}(\boldsymbol{x_0},n)$.*
It is easy to show that the test given above can be translated into a test using operations on PDBM. The formula obtained is decidable in the linear case.

The criterion to test bounded iterability of C is obtained by the conjunction of all the formula obtained for each component in $EC(C)$. If this criterion is valid, we can synthesize the boundedness constraint $T(C) \in \mathcal{B}_{SG}$ for the set $T(C)$ of transitions in C.

Example of nested loops (cont.) Let consider the symbolic graph given on Figure 5. Let C' be the connected component starting in Γ'_0, which corresponds to an odd initial value for Y_0. Let θ' be the elementary circuit of C' containing Γ'_0. The effect of C' may be computed since the elementary loops in Γ'_2 and Γ'_5 can be reduced to meta-transitions. Moreover, the effect of C' is constant and equal to $cf(\Delta_{C'}) = (0,-1,2,0)$ (see the beginning of this section). Then, the Proposition 4 can be applied. The formulas built for cases **2** and **3** above are trivially false, since all the connected sub-components of C' stop. For case **1**, the formulas built for the configurations other than Γ'_0 and Γ'_3 (i.e., for $j = 0, 3$) are also trivially false, since it is always possible to take a transition in C' from these configurations. For both configurations Γ'_0 and Γ'_3, we obtain the formula following involving the guard of the transition t_1:

$$\phi_{0,(1)}^{\hat{\theta}',\Gamma'_0}(x,y,z,n) \quad \equiv \quad (x_0,x,y,z)+n*cf(\Delta_{C'}) \not\models guard(t_1) \quad \equiv \quad x-n \leq 0$$

By Proposition 4 we have that C' is boundedly iterable iff the formula $\forall X_0, Y_0 \forall n'_1 \forall x,y,z.\ \Gamma'_0 \Longrightarrow \exists n \geq 0.\ \ x-n \leq 0$ is valid. Since in Γ'_0 we have that $x = X_0 - n'_1 - 1$, this formula is valid because for all values of X_0 and n'_1 it always exists an integer $n \geq 0$ such that $X_0 - n'_1 - 1 - n \leq 0$. So, the boundedness condition $T(C') \in \mathcal{B}_{SG}$ is generated. The same analysis can be done for the component starting in Γ''_0.

8 Conclusion

We have presented a method for verifying safety and liveness properties for fair parameterized extended automata (PEA). Our method is based on symbolic reachability analysis using powerful symbolic representation structures and extrapolation techniques. We have mainly developed techniques allowing to determine whether a component (i.e., a loop and some kind of nested loops) in the symbolic graph is boundedly iterable, i.e., corresponds to the abstraction of a terminating component of the original model. Our techniques can deal with complex systems that cannot be analyzed by the existing automatic techniques, e.g., those based on synthesis of ranking functions [12, 11]. In particular, we have shown how our techniques can be used to analyze automatically a parameterized version of BRP protocol.

References

1. P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In *Proceedings of 5th TACAS*, volume 1579 of *LNCS*. Springer Verlag, 1999.
2. P. Abdulla, A. Bouajjani, B. Jonsson, and M. Nilsson. Handling global conditions in parametrized system verification. In *Proceedings of 11th CAV*, volume 1633 of *LNCS*. Springer Verlag, 1999.
3. P.A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy, FIFO channels. In *Proceedings of the 10th CAV*, volume 1427 of *LNCS*, pages 305–317. Springer Verlag, 1998.
4. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
5. 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 CAV*, volume 1855 of *LNCS*, pages 419–434. Springer Verlag, July 2000.
6. K. Baukus, Y. Lakhnech, and K. Stahl. Verifying universal properties of parameterized networks. In *Proceedings of FTRTFT*, 2000.
7. B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of qdds. In *Proceedings of SAS'97*, volume 1302 of *LNCS*. Springer Verlag, 1997.
8. B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Proceedings of the 6th CAV*, volume 818 of *LNCS*. Springer Verlag, 1994.
9. A. Bouajjani and P. Habermehl. Symbolic reachability analysis of fifo-channel systems with nonregular sets of configurations. In *Proceedings of ICALP'97*, LNCS. Springer Verlag, 1997.
10. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th CAV*, volume 1855 of *LNCS*, pages 403–418, July 2000.
11. M. Colon and H. Sipma. Synthesis of linear ranking functions. In *Proceedings of TACAS'01*, 2001.
12. D. Dams, R. Gerth, and O. Grumberg. A heuristic for the automatic generation of ranking functions. In *Proceedings of WAVe*, pages 1–8, 2000.
13. D. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Proceedings of the 1st CAV*, volume 407 of *LNCS*, pages 197–212. Springer Verlag, 1989.
14. R.W. Floyd. Assigning meanings to programs. In *In. Proc. Symp. on Appl. Math. 19*, pages 19–32. American Mathematical Society, 1967.
15. A.C. Hearn. *REDUCE — User's and Contributed Packages Manual.* Codemist Ltd., February 1999. version 3.7.
16. Z. Manna and A. Pnueli. Completing the temporal picture. *Theoretical Computer Science*, 83(1):97–130, 1991.
17. Omega Team. *The Omega Library*, November 1996. version 1.1.0.
18. A. Pnueli and E. Shahar. Liveness and acceleration in parametrized verification. In *Proceedings of the 12th CAV*, volume 1855. Springer Verlag, 2000.