Bounded Model Checking for Parametric Timed Automata[⋆]

Michał Knapik[1] and Wojciech Penczek[1,2]

[1] Institute of Computer Science, PAS, J.K. Ordona 21, 01-237 Warszawa, Poland
{Michal.Knapik}@ipipan.waw.pl

[2] Institute of Informatics, Podlasie Academy, Sienkiewicza 51, 08-110 Siedlce, Poland
penczek@ipipan.waw.pl

Abstract. The paper shows how bounded model checking can be applied to parameter synthesis for parametric timed automata with continuous time. While it is known that the general problem is undecidable even for reachability, we show how to synthesize a part of the set of all the parameter valuations under which the given property holds in a model. The results form a complete theory which can be easily applied to parametric verification of a wide range of temporal formulae – we present such an implementation for the existential part of $\mathrm{CTL}_{-\mathrm{X}}$.

1 Introduction and related work

The growing abundance of complex systems in real world, and their presence in critical areas fuels the research in formal specification and analysis. One of the established methods in systems verification is model checking, where the system is abstracted into the algebraic model (e.g. various versions of Kripke structures, Petri nets, timed automata), and then processed with respect to the given property (usually a formula of modal or temporal logic). Classical methods have their limits however – the model is supposed to be a complete abstraction of system behaviour, with all the timing constraints explicitely specified. This situation has several drawbacks, e.g. the need to perform a batch of tests to confirm the proper system design (or find errors) is often impossible to fullfill due to the high complexity of the problem. Introducing parameters into models changes the task of *property verification* to task of *parameter synthesis*, meaning that parametric model checking tool produces the set of parameter valuations under which the given property holds instead of simple *holds/does not hold* answer. Unfortunately, the problem of parameter synthesis is shown to be undecidable for some of widely used parametric models, e.g. parametric timed automata [3, 8] and bounded parametric time Petri nets [15].

Many of model checking tools acquired new capabilities of parametric verification, e.g. UPPAAL-PMC [11] – the parametric extension of UPPAAL, LPMC [14] – extending PMC. Some of the tools were built from scratch with parametric

[⋆] Partly supported by the Polish Ministry of Science and Higher Education under the grant No. N N206 258035.

model checking in mind, e.g. TREX [1] and MOBY/DC [7]. Parametric analysis is also possible with HyTech [10] by means of hybrid automata. However, due to undecidability issues, algorithms implemented in these tools need not to stop and are very time and resource consuming. Another, very interesting approach is given in a recently developed IMITATOR tool [4] – having both the parametric timed automaton and the initial parameter valuation, IMITATOR synthesizes a set of parameter constraints. Substituting the parameters with a valuation satisfying these constraints is guaranteed to produce the timed automaton which is *time-abstract* equivalent to the one obtained from substituting the parameters with the initial valuation.

In this paper we present a new approach to parametric model checking, based on the observation that while we are not able to synthesize the *full* set of parameter constraints in general, there is no fundamental rule which forbids us from obtaining a *part* of this set. In Section 2 we introduce the parametric region graph – an extension of region graph used in theory of timed automata [2] and show (in Section 3) how the computation tree of a model can be unwinded up to some finite depth in order to apply bounded model checking (BMC) techniques [5]. To the best knowledge of the authors, this is the first application of BMC to parametric timed automata and seems to be a quite promising direction of research – firstly due to the unique BMC advantage which allows for verification of properties in limited part of the model, secondly due to the fact that it is quite easy to present BMC-based model checking algorithms for existential parts of many modal and temporal logics. In fact we describe how Parametric BMC can be implemented for the existential subset of $\mathrm{CTL}_{-\mathrm{X}}$ logic in Section 3, including the analysis of a simplified parametric model of the 4-phase handshake protocol.

2 Theory of Parametric Timed Automata

In this paper we use two kinds of variables, namely *parameters* $P = \{p_1, \ldots, p_m\}$ and *clocks* $X = \{x_0, \ldots, x_n\}$. An expression of the form $\sum_{i=1}^{m} t_i \cdot p_i + t_0$, where $t_i \in \mathbb{Z}$ is called a *linear expression.* A *simple guard* is an expression of the form $x_i - x_j \prec e$, where $i \neq j$, $\prec \in \{\leq, <\}$ and e is a linear expression. A conjunction of simple guards is called a *guard* and the set of all guards is denoted by G. We valuate the clocks in nonnegative reals, and parameters in naturals (including 0) that is $\upsilon : P \to \mathbb{N}$ is a *parameter valuation* and $\omega : X \to \mathbb{R}^{\geq 0}$ is a *clock valuation* (both υ and ω can be thought of as points in, respectively, $\mathbb{N}^m$ and ${\mathbb{R}^{\geq 0}}^n$). Additionally, following [11] we assume that $\omega(x_0) = 0$ – the "false clock" x_0 is fixed on 0 for convenience only, for uniform presentation of guards. By $e[\upsilon]$ we denote the value obtained by substituting the parameters in a linear expression e according to parameter valuation υ. We denote $\omega \models_\upsilon x_i - x_j \prec e$ iff $\omega(x_i) - \omega(x_j) \prec e[\upsilon]$ holds, and naturally extend this notion to guards. We also need a notion of *reset* that is a set of expressions of the form $x_i := b_i$ where $b_i \in \mathbb{N}$, and $0 < i \leq n$. The set of all resets is denoted by R, and the action of resetting a clock valuation ω by reset $r \in R$ is defined as following: $\omega[r]$ is a clock valuation such that $\omega[r](x_i) = b_i$ if $x_i := b_i \in r$, and $\omega[r](x_i) = \omega(x_i)$

otherwise. If $\delta \in \mathbb{R}$ and ω is a clock valuation, then $\omega + \delta$ is a clock valuation such that $(\omega + \delta)(x_i) = \omega(x_i) + \delta$ for all $0 < i \leq n$, and $\omega(x_0) = 0$. An *initial clock valuation* ω_0 is the valuation satisfying $\omega(x_i) = 0$ for all $x_i \in X$.

We also adopt a convenient notation from [11], where the $\leq$ symbol is treated as *true* and the $<$ symbol is treated as *false*. The propositional formulae built from symbols $\leq$ and $<$ are evaluated in a standard way. As to give an example, $\leq\Rightarrow<$ evaluates to $<$, $<\Rightarrow\leq$ evaluates to $\leq$, and $\neg(\leq \vee <)$ evaluates to $<$.

2.1 Parametric Timed Automata

Let us recall some notions from the theory of parametric timed automata. Non-parametric timed automata [2] are state-transition graphs augmented with a finite number of clocks, and clock constraints guarding the transitions between states. Their parametric version [3] allows for using parameters (other than clocks) in guard expressions – which may be perceived as creating the general template for system behaviour under more abstract timed constraints.

Definition 1. *A tuple $\mathcal{A} = \langle Q, q_0, A, X, P, \rightarrow, I \rangle$ where:*

- *Q is a set of* locations*,*
- *$q_0 \in Q$ is the* initial location*,*
- *A is a set of* actions*,*
- *X and P are, respectively, sets of clocks and parameters,*
- *$I : Q \rightarrow G$ is an* invariant *function,*
- *$\rightarrow \subseteq Q \times A \times G \times R \times Q$ is a transition relation.*

is called a parametric timed automaton *(PTA). All the above sets are finite. We abbreviate (q, a, g, r, q') as $q \overset{a,g,r}{\rightarrow} q'$.*

The semantics of PTA is presented below, in form of a labeled transition system.

Definition 2 (Concrete semantics). *Let $\mathcal{A} = \langle Q, q_0, A, X, P, \rightarrow, I \rangle$ be a parametric timed automaton and υ be a parameter valuation. The labeled transition system of $\mathcal{A}$ under υ is defined as a tuple $[\mathcal{A}]_\upsilon = \langle S, s_0, \overset{d}{\rightarrow} \rangle$ where:*

- *$S = \{(q, \omega) \mid q \in Q,$ and ω is a clock valuation such that $\omega \models_\upsilon I(q)\}$,*
- *$s_0 = (q_0, \omega_0)$ (we assume that $\omega_0 \models_\upsilon I(q_0)$),*
- *let $(q, \omega), (q', \omega') \in S$. The transition relation $\overset{d}{\rightarrow}$ is defined as follows:*
  - *if $d \in \mathbb{R}^{\geq 0}$, then $(q, \omega) \overset{d}{\rightarrow} (q', \omega')$ iff $q = q'$ and $\omega' = \omega + d$,*
  - *if $d \in A$, then $(q, \omega) \overset{d}{\rightarrow} (q', \omega')$ iff $q \overset{a,g,r}{\rightarrow} q'$, and $\omega \models_\upsilon g$, and $\omega' = \omega[r]$.*

The elements of S are called the concrete states *of $\mathcal{A}_\upsilon$.*

The automaton obtained by substituting parameters in the guards and the invariants of $\mathcal{A}$ by appropriate values of the parameter valuation υ is denoted by $\mathcal{A}_\upsilon$. The concrete semantics of $\mathcal{A}_\upsilon$ is defined as $[\mathcal{A}_\upsilon] = [\mathcal{A}]_\upsilon$. Notice that $\mathcal{A}_\upsilon$ is a timed automaton and $[\mathcal{A}_\upsilon]$ – its concrete semantics [2].

Our definition of parametric timed automata slightly differs from the one presented in [11], namely, we do not allow nonnegative reals as parameter values. As it was shown in [3], the choice of the parameter valuation codomain does not change the fact that the emptiness problem is undecidable. We explain the origin of this restriction in the following subsection.

2.2 Parametric Region Graph

In non-parametric timed automata theory, the *region graph* [2] is used as a part of a convenient method of presenting the concrete state space in a uniform, finite way. The finiteness of the resulting structure is a result of presence of both the *bounded* and *unbounded* regions. Intuitively, the bounded regions are convex bounded sets in the space of clock valuations, while the unbounded regions are convex and unbounded. The latter ones are defined using the maximal values of clock constraints – this is not possible in the general case of parametric timed automata (see however the optimization techniques in [11]), therefore in this paper we consider only the bounded regions. We divide the space of all the clock valuations into the set of *regions* using the following equivalence relation.

Definition 3. *Let ω, ω' be valuations of clocks $X = \{x_0, \ldots, x_n\}$. Then, $\omega \approx \omega'$ iff the following conditions hold:*

- $\lfloor \omega(x_i) \rfloor = \lfloor \omega'(x_i) \rfloor$ *for all* $x_i \in X$,
- *and* $frac(\omega(x_i)) < frac(\omega(x_j)) \iff frac(\omega'(x_i)) < frac(\omega'(x_j))$ *for all* $i \neq j, 1 \leq i, j \leq n$,
- *and* $frac(\omega(x_i)) = 0 \iff frac(\omega'(x_i)) = 0$ *for all* $x_i \in X$,

where $frac(\omega(x_i))$ denotes the fractional part of $\omega(x_i)$. The equivalence classes of $\approx$ are called (detailed) regions.

To our aims it is convenient to describe regions as sets of valuations satisfying certain guard expressions.

Lemma 1. *Let $X = \{x_0, \ldots, x_n\}$ be a set of clocks, and Z – a region of valuations. There exists a guard $g_Z = \bigwedge_{i,j \in \{0,\ldots,n\}, i \neq j} x_i - x_j \prec_{ij} b_{ij}$, such that $\prec_{ij} \in \{\leq, <\}$ and $b_{ij} \in \mathbb{Z}$ satisfying:*

$$Z = \{\omega \mid \omega \models g_Z\}.$$

Proof. We need to specify the values of b_{ij} together with the accompanying relation $\prec_{ij}$. Let $Z = [\omega]_{\approx}$ (the following considerations are valid for any choice of ω from Z).

- If $frac(\omega(x_i)) = 0$, $frac(\omega(x_j)) = 0$, let $\prec_{ij} = \leq$ and $b_{ij} = \lfloor \omega(x_i) \rfloor - \lfloor \omega(x_j) \rfloor$,
- if $frac(\omega(x_i)) \neq 0$, $frac(\omega(x_j)) = 0$, let $\prec_{ij} = <$ and $b_{ij} = \lceil \omega(x_i) \rceil - \lfloor \omega(x_j) \rfloor$,
- if $frac(\omega(x_i)) = 0$, $frac(\omega(x_j)) \neq 0$, let $\prec_{ij} = <$ and $b_{ij} = \lfloor \omega(x_i) \rfloor - \lfloor \omega(x_j) \rfloor$,
- for $frac(\omega(x_i)) \neq 0$, $frac(\omega(x_j)) \neq 0$:
  - if $frac(\omega(x_i)) = frac(\omega(x_j))$, let $\prec_{ij} = \leq$, $b_{ij} = \lfloor \omega(x_i) \rfloor - \lfloor \omega(x_j) \rfloor$,

- if $frac(\omega(x_i)) < frac(\omega(x_j))$, put $\prec_{ij} = <$, $b_{ij} = \lfloor \omega(x_i) \rfloor - \lfloor \omega(x_j) \rfloor$,
- if $frac(\omega(x_i)) > frac(\omega(x_j))$, let $\prec_{ij} = <$, $b_{ij} = \lceil \omega(x_i) \rceil - \lfloor \omega(x_j) \rfloor$.

It is easy to see that if $\omega \approx \omega'$, then for any guard g we have $\omega \models g$ iff $\omega' \models g$. Therefore, as g_Z was constructed in such a way that $\omega \models g_Z$, we have also $\omega' \models g_Z$ for all $\omega' \in Z$. On the other hand, if $\omega' \models g_Z$, then satisfaction of the guards of form $x_i - x_0 \prec_{i0} b_{i0}$ and $x_0 - x_i \prec_{0i} b_{0i}$ (recall that x_0 is fixed) guarantees that $\lfloor \omega'(x_j) \rfloor = \lfloor \omega(x_j) \rfloor$ for all $x_j \in X$. Similarly, $\omega'(x_i)$ has nonzero fractional value iff $frac(\omega(x_i)) \neq 0$, as $\omega'(x_i) \in (\lfloor \omega(x_i) \rfloor, \lceil \omega(x_i) \rceil)$, provided that $frac(\omega(x_i)) \neq 0$. Let us assume that $0 < frac(\omega(x_i))$, and $frac(\omega(x_i)) < frac(\omega(x_j))$, then from $\omega(x_i) - \omega(x_j) < \lfloor \omega(x_i) \rfloor - \lfloor \omega(x_j) \rfloor$ we have $\omega'(x_i) - \omega'(x_j) < \lfloor \omega'(x_i) \rfloor - \lfloor \omega'(x_j) \rfloor$. Therefore $\omega'(x_i) - \lfloor \omega'(x_i) \rfloor < \omega'(x_j) - \lfloor \omega'(x_j) \rfloor$, thus $frac(\omega(x_i)) < frac(\omega(x_j))$.

The guard constructed in the proof of the above lemma is called the *characteristic guard* of Z. In the above proof we used the fact that if one representative of an equivalence class satisfies a guard g, then so do all the remaining members. This is not true if we allow nonnegative reals as parameter values – for example it is easy to see that only some of representatives of class $[(0, 0.3)]$ satisfy $x_1 - x_0 < p$ under parameter valuation υ such that $\upsilon(p) = 0.5$.

Definition 4. *Let $\mathcal{A} = \langle Q, q_0, A, X, P, \rightarrow, I \rangle$ be a parametric timed automaton, $X = \{x_0, \ldots, x_n\}$ and $P = \{p_1, \ldots, p_m\}$. We introduce a relation in the set of all the pairs (Z, C) where Z is a region, and $C \subseteq \mathbb{N}^m$ is a subset of the set of all the valuations of parameters (treated as natural vectors). Let $s = x_i - x_j \prec e$ be a simple guard, and $g_Z = \bigwedge_{i,j \in \{0,\ldots,n\}, i \neq j} x_i - x_j \prec_{ij} b_{ij}$ the characteristic guard of region Z. Then we define:*

$$(Z, C) \overset{s}{\rightsquigarrow} (Z', C') \text{ iff } Z = Z' \text{ and } C' = C \cap \{\upsilon \mid b_{ij} (\prec_{ij} \Rightarrow \prec) e[\upsilon]\}.$$

Let g be a guard and s a simple guard, then:

$$(Z, C) \overset{g \wedge s}{\rightsquigarrow} (Z', C') \text{ iff for some } (Z'', C'') \text{ we have } (Z, C) \overset{g}{\rightsquigarrow} (Z'', C'')$$

$$\text{and } (Z'', C'') \overset{s}{\rightsquigarrow} (Z', C').$$

There is a natural intuition behind the above definition – if $(Z, C) \overset{g}{\rightsquigarrow} (Z', C')$ then (Z', C') contains all the pairs $(\omega, \upsilon) \in Z \times C$ such that $\omega \models_\upsilon g$. Such an operation is a counterpart for guard addition from [11], notice however that we do not need a burden of costly canonicalization. Below we state some basic properties of $\overset{g}{\rightsquigarrow}$ relation.

Lemma 2. *Let $(Z, C) \overset{g}{\rightsquigarrow} (Z', C')$, where g is a guard. Then, the following conditions hold:*

1. *if $(\omega, \upsilon) \in (Z, C)$ and $\omega \models_\upsilon g$, then $(\omega, \upsilon) \in (Z', C')$,*
2. *if $(\omega, \upsilon) \in (Z', C')$, then $\omega \models_\upsilon g$.*

Proof. Let us start with the first part of the lemma. Let us assume that $\omega \models_\upsilon g$. By the induction on the complexity of g we prove that $\upsilon \in C'$.

The base case is when $g = x_i - x_j \prec e$ (g is a simple guard). Let us assume that g_Z contains a simple guard of the form $x_i - x_j \leq b_{ij}$ where $b_{ij} \in \mathbb{Z}$. Notice that in this case the characteristic guard contains also a simple guard of the form $x_j - x_i \leq -b_{ij}$, therefore $b_{ij} = \omega'(x_i) - \omega'(x_j)$ for each $\omega' \in Z$. As $\omega \models_\upsilon g$, then $b_{ij} = \omega'(x_i) - \omega'(x_j) \prec e[\upsilon]$. Therefore $b_{ij} \prec e[\upsilon]$, which in this case means that $b_{ij}(\prec_{ij} \Rightarrow \prec)e[\upsilon]$. Now let us assume that g_Z contains a simple guard of the form $x_i - x_j < b_{ij}$. In this case, for each $\omega' \in Z$ there exists $\delta \in (0, 1)$ such that $\omega'(x_i) - \omega'(x_j) = (b_{ij} - 1) + \delta$. Let us notice that $e[\upsilon] \in \mathbb{Z}$, therefore from $(b_{ij} - 1) + \delta = \omega'(x_i) - \omega'(x_j) \prec e[\upsilon]$ we obtain $b_{ij} \leq e[\upsilon]$. The latter inequality means that in this case $b_{ij}(\prec_{ij} \Rightarrow \prec)e[\upsilon]$ holds.

For the induction step, notice that if $(Z, C) \overset{g' \wedge s}{\rightsquigarrow} (Z', C')$ (g' is a guard, and s a simple guard), then there exists (Z'', C'') such that $(Z, C) \overset{g'}{\rightsquigarrow} (Z'', C'')$ and $(Z'', C'') \overset{s}{\rightsquigarrow} (Z', C')$. From the inductive assumption we obtain that as $\omega \models_\upsilon g' \wedge s$ implies $\omega \models_\upsilon g'$, then $\upsilon \in C''$. Similarly, as $(\omega, \upsilon) \in (Z'', C'')$ and $\omega \models_\upsilon s$, we have $\upsilon \in C'$.

The proof of the second part of the lemma is also by the induction on the structure of g. Assume that $g = x_i - x_j \prec e$ and g_Z contains a simple guard of form $x_i - x_j \prec_{ij} b_{ij}$. If $(Z, C) \overset{g}{\rightsquigarrow} (Z', C')$, then $C' = C \cap \{\upsilon \mid b_{ij}(\prec_{ij} \Rightarrow \prec)e[\upsilon]\}$. As $\omega(x_i) - \omega(x_j) \prec_{ij} b_{ij}$ and $b_{ij}(\prec_{ij} \Rightarrow \prec)e[\upsilon]$ then $\omega(x_i) - \omega(x_j)(\prec_{ij} \wedge (\prec_{ij} \Rightarrow \prec))e[\upsilon]$. Therefore we have $\omega(x_i) - \omega(x_j) \prec e[\upsilon]$, thus $\omega \models_\upsilon g$.

For the induction step, let us notice that if $(Z, C) \overset{g' \wedge s}{\rightsquigarrow} (Z', C')$, then there exists (Z'', C'') such that $(Z, C) \overset{g'}{\rightsquigarrow} (Z'', C'')$ and $(Z'', C'') \overset{s}{\rightsquigarrow} (Z', C')$. If $(\omega, \upsilon) \in (Z', C')$ then by the inductive assumption $\omega \models_\upsilon s$ holds. As $C' \subseteq C'' \subseteq C$, then $\upsilon \in C''$ and $(\omega, \upsilon) \in (Z'', C'')$. Therefore, from the inductive assumption we obtain $\omega \models_\upsilon g'$ and, finally, $\omega \models_\upsilon g' \wedge s$.

From the above lemma we immediately obtain the following corollary.

Corollary 1. *Let Z be a region, and C a subset of set of all the parameter valuations. Then, the following conditions hold:*

1. *if $(Z, C) \overset{g}{\rightsquigarrow} (Z', C')$, then $Z' \times C' = Z \times C \cap \{(\omega, \upsilon) \mid \omega \models_\upsilon g\}$,*
2. *if $\omega \in Z$, $\upsilon \in C$, and $\omega \models_\upsilon g$, then $(Z, C) \overset{g}{\rightsquigarrow} (Z', C')$ for some Z', C' such that $(\omega, \upsilon) \in Z' \times C'$.*

In order to develop our theory further, we need to define two additional operations on regions.

Definition 5. *Let $Z = [\omega]_\approx$ be a region and $r \in R$ be a reset. Then,* resetting *of Z by r is defined as: $Z[r] = [\omega[r]]_\approx$.*

Clearly, resetting of a region does not depend on the choice of a representative.

Definition 6. *Let Z and Z' be two different regions. Region Z' is called a* time successor *of Z (denoted by $\tau(Z)$) iff for all $\omega \in Z$ there exists $\delta \in \mathbb{R}$ such that $\omega + \delta \in Z'$ and $\omega + \delta' \in Z \cup Z'$ for all $\delta' \leq \delta$.*

Now, we are in the position to present the notion of a *parametric region graph*, being an extension of *region graph* used in theory of timed automata [2]. The main idea is to augment regions with sets of parameter valuations under which the given concrete state (its equivalence class) is reachable from the initial state, and to mimick the transitions in the concrete semantics by their counterparts in parametric region graph.

Definition 7. *Let $\mathcal{A} = \langle Q, q_0, A, X, P, \rightarrow, I\rangle$ be a parametric timed automaton. Define the* parametric region graph *of $\mathcal{A}$ as the tuple $PREG(\mathcal{A}) = \langle S, s_0, \overset{d}{\rightarrow}\rangle$ where:*

– *$S = \{(q, Z, C) \mid q \in Q, Z$ is a region, $C \subseteq \mathbb{N}^m$ and $\forall_{\upsilon \in C} \exists_{\omega \in Z}\ \omega \models_\upsilon I(q)\}$,*
– *$s_0 = (q_0, Z_0, C_0)$ where $Z_0 = [\omega_0]_\approx$ and $C_0 = \{\upsilon \mid \omega_0 \models_\upsilon I(q_0)\}$,*
– *$(q, Z, C) \overset{d}{\rightarrow} (q', Z', C')$ is defined as follows:*
  - *if $d = \tau$ (time transition), then $q = q'$, $Z' = \tau(Z)$, and C' is such that $(Z', C) \overset{I(q)}{\rightsquigarrow} (Z', C')$,*
  - *if $d \in A$ (action transition), then there exists a transition $q \overset{d,g,r}{\rightarrow} q'$ in $\mathcal{A}$ and C'' such that $(Z, C) \overset{g}{\rightsquigarrow} (Z, C'')$ and $(Z[r], C'') \overset{I(q')}{\rightsquigarrow} (Z', C')$.*

Additionally, we call nodes of type $(q, Z, \emptyset)$ dead*, and assume that they have no outgoing transitions.*

Notice that in the above definition we could replace $\exists$ with $\forall$, due to the fact that for any guard g, fixed parameter valuation υ, and clock valuations ω, ω' such that $\omega \approx \omega'$ we have $\omega \models_\upsilon g$ iff $\omega' \models_\upsilon g$.

Both the concrete semantics of (parametric) timed automaton, and (parametric) region graph are *labelled transition systems*. We define finite and infinite runs in a labelled transition system in a usual way.

Lemma 3. *Let A be a parametric timed automaton, and $\rho_n = s_0, s_1, \ldots s_n$ a finite run in $PREG(\mathcal{A})$, where $s_i = (q_i, Z_i, C_i)$, and $C_n \neq \emptyset$. For any $(\omega, \upsilon) \in Z_n \times C_n$ there exists a finite run $\mu_n = t_0, t_1, \ldots t_n$ in $\mathcal{A}_\upsilon$, such that $t_i = (q_i, \omega_i)$, $\omega_i \in Z_i$ for $i \in \{0, \ldots, n\}$, and $\omega_n = \omega$.*

Proof. The base case of $n = 0$ is straightforward – as from the definition of $PREG(\mathcal{A})$ we have $\omega \models_\upsilon I(q_0)$ for any $(\omega, \upsilon) \in Z_0 \times C_0$.

Recall that $C_n \subseteq C_{n-1}$. If $s_{n-1} \overset{d}{\rightarrow} s_n$ is a time transition (with $d = \tau$), then $\tau(Z_{n-1}) = Z_n$. Therefore for each $\omega_n \in Z_n$ there exist $\omega_{n-1} \in Z_{n-1}$, and $l \in \mathbb{R}$, such that $\omega_n = \omega_{n-1} + l$. We conclude the case by noticing that $(\omega_{n-1}, \upsilon) \in Z_{n-1} \times C_{n-1}$, $\omega_n \models_\upsilon I(q_n)$, and using the inductive assumption.

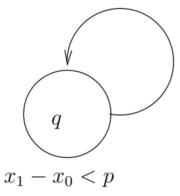
Now, if $s_{n-1} \overset{d}{\rightarrow} s_n$ is an action transition ($d \in A$), then there exists a transition $q_{n-1} \overset{d,g,r}{\rightarrow} q_n$ in $\mathcal{A}$, and a subset C' of $\mathbb{N}^m$, such that $(Z_{n-1}, C_{n-1}) \overset{g}{\rightsquigarrow}$

(Z_{n-1}, C'), and $(Z_{n-1}[r], C') \overset{I(q_n)}{\rightsquigarrow} (Z_{n-1}[r], C_n)$. Therefore for each $\omega_n \in Z_n$ we have $\omega_n \models_\upsilon I(q_n)$, and there exists $\omega_{n-1} \in Z_{n-1}$ such that $\omega_n = \omega_{n-1}[r]$, $\omega_{n-1} \models_\upsilon I(q_{n-1})$, and $\omega_{n-1} \models_\upsilon g$ (notice that $\upsilon \in C_n \cap C' \cap C_{n-1}$). We conclude the case by assuming $t_{n-1} = (q_{n-1}, \omega_{n-1})$, $t_n = (q_n, \omega_n)$ and using the inductive assumption.

Notice that the definition of the transition relation in $PREG(\mathcal{A})$ implies that in ρ_n we have $C_{i+1} \subseteq C_i$ for all $0 \leq i < n$. In particular $C_n \subseteq C_i$ for all $0 \leq i \leq n$.

The above lemma does not extend to infinite runs, as shown in the following example.

Example 1. Consider the simple parametric timed automaton:



The following infinite run in $PREG(\mathcal{A})$ does not have a counterpart in $\mathcal{A}_\upsilon$ due to the fact that p is unbounded.

$$(q, [(0,0)], \{p \mid p > 0\}) \xrightarrow{\tau} (q, [(0, 0.1)], \{p \mid p \geq 1\}) \xrightarrow{\tau}$$

$$(q, [(0,1)], \{p \mid p > 1\}) \xrightarrow{\tau} (q, [(0, 1.1)], \{p \mid p \geq 2\}) \xrightarrow{\tau} \ldots$$

Consider a transition $(q, Z, C) \xrightarrow{d} (q', Z', C')$ in $PREG(\mathcal{A})$. Notice that if $\omega \in Z$, $\upsilon \in C \cap C'$, then $(q, \omega) \xrightarrow{d'} (q', \omega')$ in $[\mathcal{A}_\upsilon]$, where $d' = d$ if d is an action, and d' is some real number if $d = \tau$. From this observation and Lemma 3 we obtain the following corollary.

Corollary 2. *Let $\rho = s_0, s_1, \ldots$ be an infinite run in $PREG(\mathcal{A})$, such that $s_i = (q_i, Z_i, C_i)$ for some Z_i, C_i, and let $\upsilon \in C_i$ for all $i \geq 0$. Then, there exists an infinite run $\mu = t_0, t_1, \ldots$ in the concrete semantics of $\mathcal{A}_\upsilon$, such that $t_i = (q_i, \omega_i)$, and $\omega_i \in Z_i$.*

The counterpart of Lemma 3 holds without the restriction on finiteness of runs.

Lemma 4. *Let $\mathcal{A}$ be a parametric timed automaton, and $\mu = t_0, t_1, \ldots t_n \ldots$ an infinite (finite) run in $\mathcal{A}_\upsilon$, where $t_i = (q_i, \omega_i)$, and such that if $t_i \xrightarrow{d} t_{i+1}$ is a time transition, then $[\omega_{i+1}] = \tau([\omega_i])$. Then, there exists an infinite (finite, resp.) run $\rho = s_0, s_1, \ldots s_n \ldots$ in $PREG(\mathcal{A})$ such that $s_i = (q_i, Z_i, C_i)$, and $(\omega_i, \upsilon) \in Z_i \times C_i$ for each $i \geq 0$ ($0 \leq i \leq n$, resp.).*

Proof. Let us start with the finite run case, and let $Z_i = [\omega_i]$. The base case is straightforward – just assume $C_0 = \{u \mid \omega_0 \models_u I(q_0)\}$ and notice that $\upsilon \in C_0$.

Assume that we have already constructed a finite run $\rho_n = s_0, s_1, \ldots s_{n-1}$.

If $t_{n-1} \overset{d}{\to} t_n$ is a time transition, then $\tau(Z_{n-1}) = Z_n$, $\omega_n \in Z_n$, $\upsilon \in C_{n-1}$, and $\omega_n \models_\upsilon I(q_n)$. Therefore, from Corollary 1 we obtain that there exists C' such that $(Z_n, C_{n-1}) \overset{I(q_n)}{\leadsto} (Z_n, C')$, $\upsilon \in C'$, and conclude the case by placing $C_n = C'$, and the inductive assumption.

If $t_{n-1} \overset{d}{\to} t_n$ is an action transition, then there exists a transition in $\mathcal{A}$ such that for some guard g and reset r we have $q_{n-1} \overset{d,g,r}{\to} q_n$. Notice that as $(\omega_{n-1}, \upsilon) \in Z_{n-1} \times C_{n-1}$, $\omega_{n-1} \models_\upsilon g$, $\omega_{n-1}[r] = \omega_n$, and $\omega_n \models_\upsilon I(q_n)$, from Corollary 1 we have that there exist sets C', C'' satisfying $(Z_{n-1}, C_{n-1}) \overset{g}{\leadsto} (Z_{n-1}, C')$, $\upsilon \in C'$, and $(Z_{n-1}, C') \overset{I(q_n)}{\leadsto} (Z_n, C'')$. We conclude the case by assuming $C_n = C''$.

Let $\mu = t_0, t_1, \ldots$ be an infinite run in $\mathcal{A}_\upsilon$. We have already shown that for each finite prefix $\mu_n = t_0, t_1, \ldots t_n$ we can construct its counterpart $\rho_n = s_0^n, s_1^n, \ldots s_n^n$ in $PREG(\mathcal{A})$, where $s_n^i = (q_i, Z_i, C_i^n)$. Notice that $C_i^n = C_i^{n+1}$, so the infinite sequence $\rho = s_0, s_1, \ldots$, where $s_i = (q_i, Z_i, C_i^i)$ is a valid infinite run in $PREG(\mathcal{A})$ satisfying $(\omega_i, \upsilon) \in Z_i \times C_i^i$ for all $i \geq 0$.

The following definition formalizes the connection between parametric region graph, and region graphs. In what follows, by a subgraph of $PREG(\mathcal{A}) = \langle S, s_0, \overset{d}{\to}\rangle$ we mean a tuple $\langle S', s_0, \overset{d}{\hookrightarrow}\rangle$, where S' is a subset of S, and $\overset{d}{\hookrightarrow}$ is the restriction of $\overset{d}{\to}$ to S'.

Definition 8. *Let $\mathcal{A}$ be a parametric timed automaton, υ – a parameter valuation, and F – a subgraph of $PREG(\mathcal{A})$. By $proj(F, \upsilon)$ we define a subgraph of F whose states are tuples (q, Z, C) such that $\upsilon \in C$.*

Observe that $proj(PREG(\mathcal{A}), \upsilon)$ is in fact isomorphic with the region graph of $\mathcal{A}_\upsilon$ – by a forgetful functor stripping C from tuple (q, Z, C).

3 Bounded Model Checking for $\mathbf{ECTL_{-X}}$

The central idea of bounded model checking is to unfold the computation tree of a considered model up to some depth, and then perform the analysis of such a finite structure [5]. Such an approach limits us to verification (and in our case – parameter synthesis) of existential properties only, it should be noted however that implicit model checking methods often fail in case of large and complex systems. Bounded model checking seems to be especially effective in searching for counterexamples, i.e. in proving that some undesirable property holds in a model. This allows for detection of serious design flaws of concurrent and reactive systems.

The non-parametric model checking tool verifies a model (system specification) against a given property (usually in form of a temporal logic formula), producing the answer of simple *holds/does not hold* type. Its parametric counterpart is supposed to work slightly differently – having a parametric model we expect the answer in form of a set of parameter values under which a given property is satisfied. The automated synthesis of a complete set of desired parameter valuations is not possible in case of timed automata due to general undecidability of the problem, however obtaining a part of this set still seems to be a worthy goal. Our approach allows for incremental synthesis of parameters, i.e. if the valuations obtained by analysis of a part of a computation tree are not sufficient, then the tree can be unfolded up to a greater depth for further analysis. Combined with an expert supervision, the synthesized parameter valuations can give rise to hypotheses specifying the whole space of desired parameters.

We propose the following general flow of property verification/parameter synthesis.

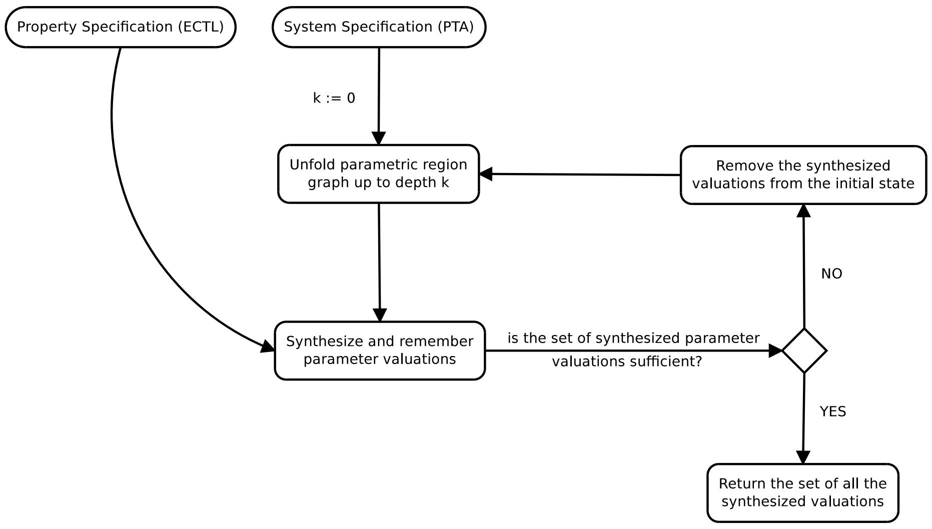

Property Specification (ECTL)
System Specification (PTA)
k := 0
Unfold parametric region graph up to depth k
Remove the synthesized valuations from the initial state
NO
Synthesize and remember parameter valuations
is the set of synthesized parameter valuations sufficient?
YES
Return the set of all the synthesized valuations

Fig. 1. Parametric Bounded Model Checking schema

The above diagram is very general. One of the approaches in the current applications of bounded model checking to verification of system properties is to encode the limited part of the computation tree together with a property in question as a propositional formula [6, 13]. The result can be checked using an efficient SAT-solver.

3.1 From Parametric Region Graph to concrete semantics

The $PREG(\mathcal{A})$ structure is infinite. In order to represent the infinite runs in a finite substructure we need a notion of *loop*.

Definition 9. *Let $\rho_n = s_0, s_1, \ldots s_n$ be a finite run in $PREG(\mathcal{A})$, and $s_i \overset{d}{\to} s_{i+1}$ for all $0 \le i < n$. If $s_n = (q_n, Z_n, C_n)$ and there exists $s_i = (q_i, Z_i, C_i)$, where $0 \le i < n$ such that $s_n \overset{d}{\to} s_i$ and $q_n = q_i$, $Z_n = Z_i$, then ρ_n is called a* loop.

Let $\rho_n = s_0, s_1, \ldots s_n$ be a loop in $PREG(\mathcal{A})$, such that $s_i = (q_i, Z_i, C_i)$, and $(q_n, Z_n) = (q_j, Z_j)$ for some $j < n$. We can create an infinite run $\hat{\rho} = \hat{s_0}, \hat{s_1}, \ldots$ by unwinding the ρ_n loop as follows:

$$\hat{s_i} = \begin{cases} (q_i, Z_i, C_n) & \text{for } i < n \\ (q_{j+(n-i)mod(n-j)}, Z_{j+(n-i)mod(n-j)}, C_n) & \text{for } i \ge n. \end{cases}$$

The validity of such a construction is based on the observation that $C_n \subseteq C_i$ for all $0 \le i \le n$ and the fact that transitions in $PREG(\mathcal{A})$ are defined in terms of g_Z and guards only. Applying Corollary 2 to such an unwinding we obtain the following corollary.

Corollary 3. *Let $\rho = s_0, s_1, \ldots, s_n$ be a loop in $PREG(\mathcal{A})$, where $s_i = (q_i, Z_i, C_i)$, and $\upsilon \in C_n$ – a parameter valuation. There exists an infinite run $\mu_t = t_0, t_1, \ldots$ in the concrete semantics of $\mathcal{A}_\upsilon$, where $t_i = (\hat{q_i}, \omega_i)$, $\omega_i \in Z_i$ for $i < n$, $\omega_i \in Z_{j+(n-i)mod(n-j)}$ for $i \ge n$, and:*

$$\hat{q_i} = \begin{cases} q_i & \textit{for } i < n \\ q_{j+(n-i)mod(n-j)} & \textit{for } i \ge n. \end{cases}$$

3.2 Parametric Bounded Model Checking for ECTL$_{-\mathrm{X}}$

The presented method can be applied to the verification of a variety of properties. As the example, in this subsection we present the application of introduced theory to verification of properties specified in the existential part of Computation Tree Logic (CTL$_{-\mathrm{X}}$) without the *next* operator [9] – namely ECTL$_{-\mathrm{X}}$. Intuitively, CTL$_{-\mathrm{X}}$ uses a branching time model, where many possible paths in the future exist. The whole CTL$_{-\mathrm{X}}$ contains both the universal ("for all the possible paths") and existential modalities ("there exists a path in the future") while ECTL$_{-\mathrm{X}}$ contains only the latter ones – see [13] for more thorough treatment.

Definition 10 (CTL$_{-\mathrm{X}}$ **and** ECTL$_{-\mathrm{X}}$ **syntax).** *Let $\mathcal{PV}$ be a set of propositions containing the* true *symbol, and $p \in \mathcal{PV}$. The set of well-formed* CTL$_{-\mathrm{X}}$ *formulae is given by the following grammar:*

$$\Phi ::= p \mid \neg\Phi \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid EG\Phi \mid E\Phi U\Phi.$$

The existential subset of CTL$_{-\mathrm{X}}$*, i.e.* ECTL$_{-\mathrm{X}}$ *is defined as a restriction of* CTL$_{-\mathrm{X}}$ *such that the negation can be applied to the propositions only.*

Additionally we use the derived modalities: $EF\alpha \overset{def}{=} E(trueU\alpha)$, $AF\alpha \overset{def}{=} \neg EG\neg\alpha$, $AG\alpha \overset{def}{=} \neg EF\neg\alpha$. Each modality of CTL$_{-\mathrm{X}}$ has an intuitive meaning. The path quantifier A stands for "on every path" and E means "there exists a path". G stands for "in all the states", F means "in some state", and U has a meaning of "until".

We augment the given parametric timed automaton $\mathcal{A} = \langle Q, q_0, A, X, P, \rightarrow, I\rangle$ with a labelling function $\mathcal{L} : Q \rightarrow 2^{\mathcal{PV}}$. Let us present an intepretation of $\mathrm{ECTL}_{-\mathrm{X}}$ formulae for a parametric region graph.

Definition 11 **($\mathrm{ECTL}_{-\mathrm{X}}$ semantics for parametric region graph).** *Let $\mathcal{A} = \langle Q, q_0, A, X, P, \rightarrow, I\rangle$ be a parametric timed automaton, and F – a subgraph of its parametric region graph, such that (q_0, Z_0, C'_0), where $C'_0 \subseteq C_0$, is a state of F. Let s be a state of F, $p \in \mathcal{PV}$, and α, β be $\mathrm{ECTL}_{-\mathrm{X}}$ formulae. We treat F as a model for $\mathrm{ECTL}_{-\mathrm{X}}$ formulae, defining the $\models$ relation as follows.*

1. *$F, (q, Z, C) \models p$ iff $p \in \mathcal{L}(q)$,*
2. *$F, s \models \neg p$ iff $F, s \not\models p$,*
3. *$F, s \models \alpha \vee \beta$ iff $F, s \models \alpha$ or $F, s \models \beta$,*
4. *$F, s \models E\alpha U\beta$ iff there exists a run $\rho_n = s_0, s_1, \ldots$, where $s_0 = s$, s_i are states of F for $i \geq 0$, $F, s_j \models \beta$ for some $j \geq 0$, and $F, s_i \models \beta$ for all $i < j$,*
5. *$F, s \models EG\alpha$ iff there exists a run $\rho_n = s_0, s_1, \ldots$, such that $F, s_i \models \alpha$ for all $i \geq 0$.*

We abbreviate $F, (q_0, Z_0, C_0) \models \alpha$ as $F \models \alpha$.

The counterpart of the above definition for the timed automaton $\mathcal{A}_\upsilon = \langle S, s_0, \xrightarrow{d}\rangle$ obtained from the parametric timed automaton $\mathcal{A}$ under the parameter valuation υ is similar – except for that it is defined over the concrete semantics ($s \in S$). Therefore the only difference is in the first clause which takes the following form:

1. *$\mathcal{A}_\upsilon, (q, \omega) \models p$ iff $p \in \mathcal{L}(q)$*

As previously, we abbreviate $\mathcal{A}_\upsilon, (q_0, \omega_0) \models \alpha$ as $\mathcal{A}_\upsilon \models \alpha$.

In order to apply bounded model checking to verification of temporal properties in $PREG(\mathcal{A})$ we need to specify the version of the above semantics for finite subgraphs of $PREG(\mathcal{A})$. The only difference concerns clauses 4 and 5 which take the following form:

4. *$F, s \models E\alpha U\beta$ iff there exists a finite run $\rho_n = s_0, s_1, \ldots s_n$, where $s_0 = s$, s_i are states of F for $0 \leq i \leq n$, $F, s_j \models \beta$ for some $0 \leq j \leq n$, and $F, s_i \models \beta$ for all $i < j$,*
5. *$F, s \models EG\alpha$ iff there exists a loop $\rho_n = s_0, s_1, \ldots, s_n$, such that $F, s_i \models \alpha$ for all $0 \leq i \leq n$.*

Recall that timed automaton $\mathcal{A}_\upsilon$ is *strongly non-zeno* (see [16]) iff for each sequence of states $q_1, \ldots, q_n$ such that $q_i \xrightarrow{a_i, g_i, r_i} q_{i+1}$ for all $0 \leq i < n$, and $q_n \xrightarrow{a_n, g_n, r_n} q_1$ (we call such a sequence a *structural loop*) there exists a clock x satisfying the following conditions:

– for some $1 \leq i \leq n$ the x clock is reset in step i (i.e. $x := 0 \in r_i$),

– there exists $1 \leq j \leq n$ such that for any clock valuation ω if $\omega \models_{\upsilon} g_j$, then $\omega(x) \geq 1$.

Intuitively, if an automaton is strongly non-zeno, then in each its loop at least one unit of time elapses ([16]). Notice that checking if the automaton is strongly non-zeno does not require any representation of the state space.

Theorem 1. *Let $\mathcal{A}$ be a parametric timed automaton, F – a finite subgraph of $PREG(A)$ containing state (q_0, Z_0, C_0'), where $C_0' \subseteq C_0$, and $P = \bigcap\{C \mid (q, Z, C)$ is a state of $F\}$. If P is nonempty, and $\mathcal{A}_\upsilon$ is strongly non-zeno for each $\upsilon \in P$, then for each formula $\alpha \in \mathrm{ECTL}_{-\mathrm{X}}$ if $F \models \alpha$, then $\mathcal{A}_\upsilon \models \alpha$ for all $\upsilon \in P$.*

Proof. Let $\upsilon \in P$ be a parameter valuation. Denote by $\hat{F}$ a (possibly infinite) subgraph of $PREG(\mathcal{A})$ created in two steps:

– firstly, by adding to F the new states created by unwinding of each loop along the lines presented above – obtaining F',
– secondly, by replacing all the states (q, Z, C) in F' by (q, Z, P) – obtaining $\hat{F}$.

It is easy to see that $F \models \alpha$ iff $\hat{F} \models \alpha$. Recall that $proj(\hat{F}, \upsilon)$ is isomorphic to some subgraph of the region graph of $\mathcal{A}_\upsilon$. As satisfiability of $\mathrm{ECTL}_{-\mathrm{X}}$ formulae in a subgraph of the region graph implies satisfiability in the region graph, and satisfiability in region graph is equivalent to satisfiability in the concrete model (see [16]) we obtain the thesis of the theorem.

3.3 Example – four phase handshake protocol

In this section we perform a first step in parametric analysis of a simplified version of four phase handshake protocol. The protocol is extensively used in practice and widely studied, having both the software and hardware implementations [**?**,**?**]. The considered system consists of two communicating entities – the Producer and the Consumer. The Producer creates data packages and sends them to the Consumer. Both the components communicate using two shared boolean variables, that is: **req** (request) governed by the Producer and used to signal the Consumer that the data is prepared and ready to be read, and **ack** (acknowledge) governed by the Consumer and used to signal the Producer that the data has been read successfully and the Consumer is ready. The initial value of both the variables is *false*.

The running system goes through the following sequence of signals (req, ack):

$$(false, false) \to (true, false) \to (true, true) \to (false, true) \to (false, false).$$

As we have no tool for automated analysis at our disposal yet, we analyze the simplified version of the system behaviour. We introduce two parameters, omitting the signal propagation time, namely: $minIO$, and $maxIO$ being, respectively, the lower and the upper bound on read/write time.

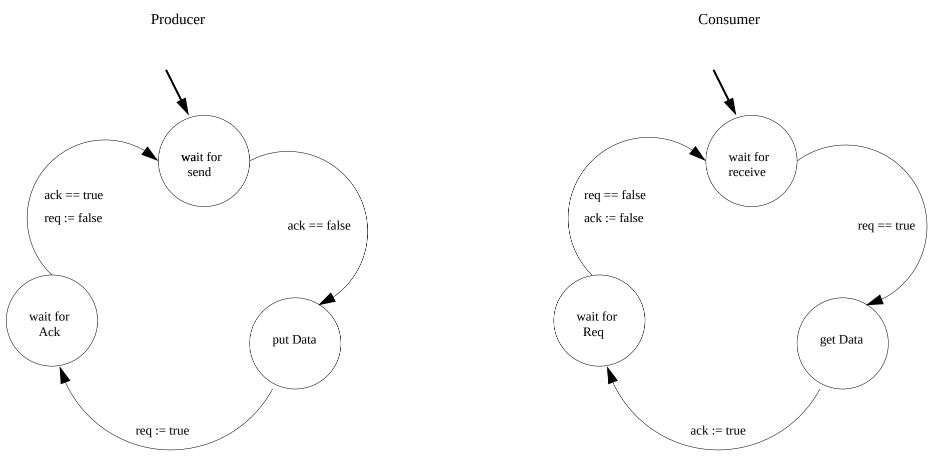


Fig. 2. 4–phase handshake protocol

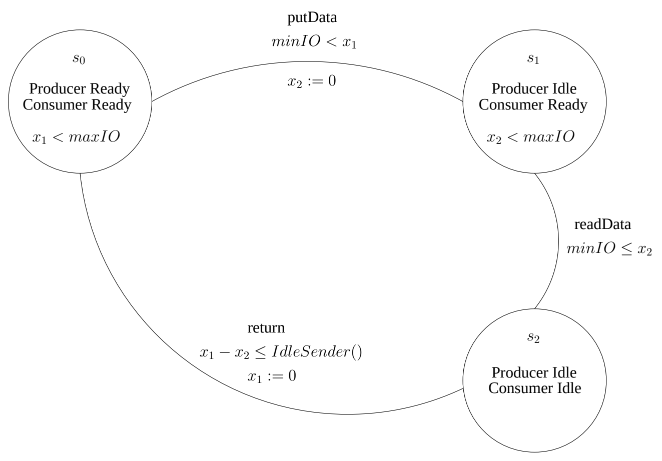
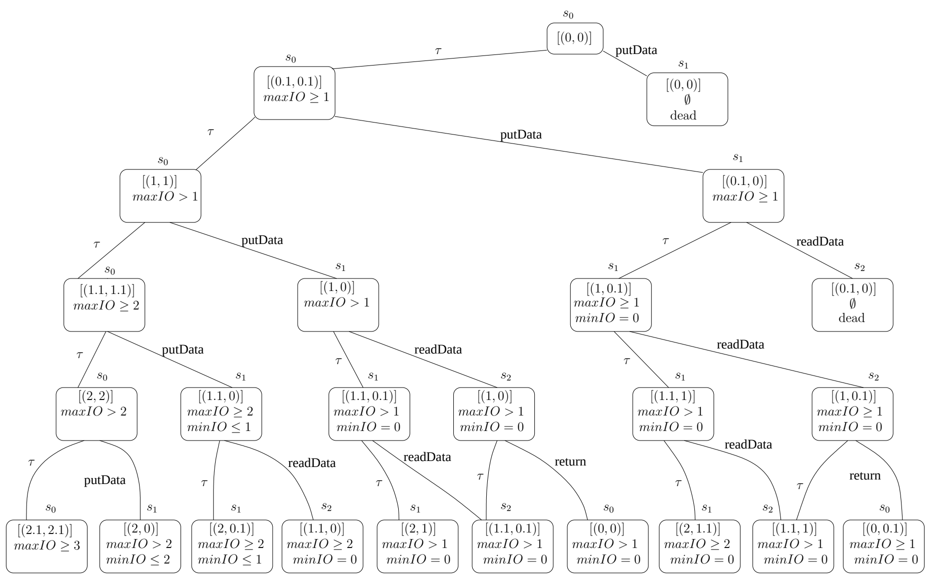

s_0
Producer Ready Consumer Ready
$x_1 < maxIO$
putData
$minIO < x_1$
$x_2 := 0$
s_1
Producer Idle Consumer Ready
$x_2 < maxIO$
readData
$minIO \leq x_2$
s_2
Producer Idle Consumer Idle
return
$x_1 - x_2 \leq IdleSender()$
$x_1 := 0$

Fig. 3. 4–phase handshake protocol, behaviour diagram

The $IdleSender$ function guards the time that the Producer is allowed to be idle after putting data into some shared transmission vehicle (e.g. a bus). Let us put $IdleSender() := maxIO - minIO$ and unwind the Parametric Region Graph of Figure 3 (we omit the dummy clock x_0).

Notice that the above graph contains a loop, introduced by the sequence of actions: $\tau, \tau, putData, readData, return$. This loop can be unwinded as presented in Subsection 4.1 into an infinite path in the Parametric Region Graph, and into



s_0 $[(0,0)]$
τ
putData
s_1 $[(0,0)]$ $\emptyset$ dead
s_0 $[(0.1,0.1)]$ $maxIO \geq 1$
τ
putData
s_0 $[(1,1)]$ $maxIO > 1$
s_1 $[(0.1,0)]$ $maxIO \geq 1$
τ
putData
τ
readData
s_0 $[(1.1,1.1)]$ $maxIO \geq 2$
s_1 $[(1,0)]$ $maxIO > 1$
s_1 $[(1,0.1)]$ $maxIO \geq 1$ $minIO = 0$
s_2 $[(0.1,0)]$ $\emptyset$ dead
τ
putData
τ
readData
τ
readData
s_0 $[(2,2)]$ $maxIO > 2$
s_1 $[(1.1,0)]$ $maxIO \geq 2$ $minIO \leq 1$
s_1 $[(1.1,0.1)]$ $maxIO > 1$ $minIO = 0$
s_2 $[(1,0)]$ $maxIO > 1$ $minIO = 0$
s_1 $[(1.1,1)]$ $maxIO > 1$ $minIO = 0$
s_2 $[(1,0.1)]$ $maxIO \geq 1$ $minIO = 0$
τ
putData
τ
readData
τ
readData
τ
return
τ
readData
τ
return
s_0 $[(2.1,2.1)]$ $maxIO \geq 3$
s_1 $[(2,0)]$ $maxIO > 2$ $minIO \leq 2$
s_1 $[(2,0.1)]$ $maxIO \geq 2$ $minIO \leq 1$
s_2 $[(1.1,0)]$ $maxIO \geq 2$ $minIO = 0$
s_1 $[(2,1)]$ $maxIO > 1$ $minIO = 0$
s_2 $[(1.1,0.1)]$ $maxIO > 1$ $minIO = 0$
s_0 $[(0,0)]$ $maxIO > 1$ $minIO = 0$
s_1 $[(2,1.1)]$ $maxIO \geq 2$ $minIO = 0$
s_2 $[(1.1,1)]$ $maxIO > 1$ $minIO = 0$
s_0 $[(0,0.1)]$ $maxIO \geq 1$ $minIO = 0$

Fig. 4. The 4–phase handshake protocol, Parametric Region Graph of depth 5

loops in concrete semantics of non-parametric timed automata with $minIO = 0$, and $maxIO$ instantiated by any value greater that 1.

The graph of Figure 4, treated as a subgraph of the Parametric Region Graph of Figure 3 allows us to observe that in the considered system the property $EGEF(ProducerIdle \wedge ConsumerReady)$ holds for $minIO = 0$, and $maxIO > 1$, with the previously mentioned loop as a witness. The intuition behind the considered formula is that the Producer will put data into the transmission infinitely often in the running system.

Of course, this is only the first, hand-made, step of synthesis of the parameter valuations under which the considered property is satisfied. The complete analysis of non-simplified versions with more parameters and components has to wait until we develop the planned tool.

4 Future work

The theory presented in this paper is to be implemented in Verics model checker [12]. There is a growing evidence [14, **?**] of success of model checking in verification of safety critical industrial applications, and the idea of parameter synthesis for a complex model or protocol seems to be promising in analysis and design of real-world systems. Also, as the method is quite general, we expect that it may be applied to many known temporal, modal and epistemic logics.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur, T. Henzinger, and M. Vardi. Parametric real-time reasoning. In *Proc. of the 25th Ann. Symp. on Theory of Computing (STOC'93)*, pages 592–601. ACM, 1993.
3. É. André, T. Chatain, E. Encrenaz, and L. Fribourg. An inverse method for parametric timed automata. *International Journal of Foundations of Computer Science*, 20(5):819–836, October 2009.
4. A. Annichini, A. Bouajjani, and M. Sighireanu. Trex: A tool for reachability analysis of complex systems. In *Proc. of CAV*, pages 368–372, 2001.
5. A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. In *Highly Dependable Software*, volume 58 of *Advances in Computers*. Academic Press, 2003. Pre-print.
6. I. Blunno, J. Cortadella, A. Kondratyev, L. Lavagno, K. Lwin, and C. Sotiriou. Handshake protocols for de-synchronization. In *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 149–158. IEEE Computer Society Press, 2004.
7. E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.
8. H. Dierks and J. Tapken. Moby/DC - a tool for model-checking parametric real-time specifications. In *Proc. of the 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *LNCS*, pages 271–277. Springer-Verlag, 2003.
9. L. Doyen. Robust parametric reachability for timed automata. *Information Processing Letters*, 102:208–213, 2007.
10. E. A. Emerson and E. Clarke. Using branching-time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.
11. S.B. Furber and P. Day. Four-phase micropipeline latch control circuits. In *IEEE Transactions on VLSI Systems*, volume 4, pages 247–253, 1996.
12. T. Henzinger, P. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. In *Proc. of the 9th Int. Conf. on Computer Aided Verification (CAV'97)*, volume 1254 of *LNCS*, pages 460–463. Springer-Verlag, 1997.
13. T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager. Linear parametric model checking of timed automata. In *Proc. of the 7th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'01)*, volume 2031 of *LNCS*, pages 189–203. Springer-Verlag, 2001.
14. M. Kacprzak, W. Nabiałek, A. Niewiadomski, W. Penczek, A. Półrola, M. Szreter, B. Woźna, and A. Zbrzezny. VerICS 2008 - a model checker for time Petri nets and high-level languages. In *Proc. of Int. Workshop on Petri Nets and Software Engineering (PNSE'09)*, pages 119–132. University of Hamburg, 2009.
15. Spelberg R. L., De Rooij R. C. H., , and Toetenel W. J. Application of parametric model checking – the root contention protocol using lpmc. In *Proc. of the 7th ASCI Conference*, pages 73–85, February 2000.
16. W. Penczek, B. Woźna, and A. Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fundamenta Informaticae*, 51(1-2):135–156, 2002.
17. M. Stoelinga. Fun with firewire: A comparative study of formal verification methods applied to the ieee 1394 root contention protocol. In *Formal Asp. Comput.*, volume 14, pages 328–337, 2003.

18. L-M. Tranouez, D. Lime, and O. H. Roux. Parametric model checking of time Petri nets with stopwatches using the state-class graph. In *Proc. of the 6th Int. Workshop on Formal Analysis and Modeling of Timed Systems (FORMATS'08)*, volume 5215 of *LNCS*, pages 280–294. Springer-Verlag, 2008.
19. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.