Compositional Parameter Synthesis

Lacramioara Astefanoaiei, Saddek Bensalem, Marius Bozga, Chih-Hong Cheng, Harald Ruess

Compositional Parameter Synthesis

L. Aştefănoaei[1], S. Bensalem[2], M. Bozga[2], C.-H. Cheng[1], and H. Ruess[1]

[1] fortiss - An-Institut Technische Universität München**
[2] Univ. Grenoble Alpes, VERIMAG, F-38000 Grenoble, France

Abstract. We address the problem of parameter synthesis for parametric timed systems (PTS). The motivation comes from industrial configuration problems for production lines. Our method consists in compositionally generating over-approximations for the individual components of the input systems, which are translated, together with global properties, to ∃∀SMT problems. Our translation forms the basis for optimised and robust parameter synthesis for slightly richer models than PTS.

1 Introduction

Synthesis for parametric timed automata (PTA) has drawn considerable attention [1, 19, 20, 26, 17, 14, 15, 2, 3, 25, 12, 7, 10, 18, 21, 13]. These approaches explore the *global* state space of all interacting components. In contrast, our method is compositional, consequently, in this regard, it scales well to large systems.

Our motivation comes from parameter configuration problems for production lines such as the ones from the food sector described in [8]. Seeing the constituting machines as interacting PTAs, configuration problems fit well the class of systems we study. Concretely, our contribution is to show how, given (1) a *parametric timed system* $\mathcal{S}$ with unknown parameters p, (2) constraints ϕ_p on p, and (3) a safety property ϕ_{safe} for $\mathcal{S}$, we automatically generate, in a compositional manner and by means of an ∃∀SMT solver, valuations for p such that the desired safety property holds. In particular, we reduce the parameter synthesis problem to solving formulae of the type:

$$\exists p \in \phi_p . \forall v . \big(\psi_{\mathcal{S}}(p, v) \rightarrow \phi_{safe}\big) \tag{1}$$

where v represents all other variables (clocks, locations) except p and $\psi_{\mathcal{S}}(p, v)$ is an over-approximation of the behaviour of $\mathcal{S}$. A PTS is composed of components (PTAs) interacting by multi-party interactions. Given n components C_i and interactions γ, $\|_\gamma C_i$ denotes the corresponding PTS. To compute $\psi_{\mathcal{S}}$ for $\mathcal{S} \triangleq \|_\gamma C_i$ we adapt and extend the methodology from [4] to the parametric setup. We first equip each component C_i with history clocks. Let C_i^h be the results. We then compute three types of invariants: (1) interaction invariant from γ; (2) component invariants from the parametric zones of C_i^h; and (3) relations between history clocks. In [4], history clocks were used to derive relations between clocks in different components. In the parametric case, history clocks are used to also derive relations on parameters. This helps synthesise parameters which, for instance, do not introduce deadlock in the system.

2 Parametric Timed Systems and Properties

A valuation $\mathbf{v}$ is a function that assigns a real value $\mathbf{v}(x)$ to each variable x. A linear inequality has the form $\sum_{i=1}^{n} \alpha_i x_i \# \beta$ with x_i being variables, $\alpha_i, \beta \in \mathbb{Z}$, $\# \in \{<, \leq, =, \geq, >\}$. A convex linear constraint is a finite conjunction of linear inequalities. The set of convex linear constraints over a set of variables $\mathcal{V}$ is denoted by $\mathcal{L}(\mathcal{V})$.

** Work supported by the European projects BEinCPPS, CPSE-labs and OpenMOS.

Definition 1. *A component is a PTA* $(L, l_0, \mathcal{X}, \mathcal{P}, A, T, \mathsf{tpc})$ *where:* l_0 *is an initial location;* $L, \mathcal{X}, \mathcal{P}, A, T$ *are finite sets of locations, clock variables, parameters (variables whose values do not change over time), actions, and transitions. Transitions* $l \xrightarrow{a,g,\mu} l'$ *consist of a source* $l \in L$ *and a target location* $l' \in L$*, an action* $a \in A$*, a guard condition* g *in* $\mathcal{L}(\mathcal{X} \cup \mathcal{P})$*, and a jump relation* $\mu \in \mathcal{L}(\mathcal{X} \cup \mathcal{X}')$ *with* $\mathcal{X}'$ *denoting the clocks at* l'*.* $\mathsf{tpc} : L \rightarrow \mathcal{L}(\mathcal{X} \cup \mathcal{P})$ *assigns convex linear clock constraints to locations.*

For a parameter valuation $\mathbf{v}$ and a component C, the concrete semantics of C under $\mathbf{v}$, $C(\mathbf{v})$, is that of a timed automaton. Since this semantics yields an infinite state space, we work with *parametric zone graphs* as finite symbolic representations. The symbolic states in a parametric zone graph are pairs (l, ζ) of a location l and a convex linear constraint ζ over clocks and parameters which can be represented by convex polyhedra [20]. For $C \triangleq (L, l_0, \mathcal{X}, \mathcal{P}, A, T, \mathsf{tpc})$ the parametric zone graph is computed from T starting from an initial symbolic state $(l_0, \mathbf{v}_0)$, with $\mathbf{v}_0(x) = 0$ for all $x \in \mathcal{X}$, and using the successor operator. For a transition t, the successor operator of (l, ζ) is defined as $\mathsf{succ}(t, (l, \zeta)) \triangleq \mathsf{time\_succ}(\mathsf{disc\_succ}(t, (l, \zeta)))$ where $\mathsf{disc\_succ}$. resp. $\mathsf{time\_succ}$ are the discrete, resp. the time successor. We recall their definitions from [21]. The operation $\mathsf{time\_succ}$ for letting time progress within a symbolic state is defined as $\mathsf{time\_succ}((l, \zeta)) \triangleq (l, \zeta^{\nearrow})$ where $^{\nearrow}$ is the time-elapse operator. The successor with respect to a transition $t \triangleq (l, (\_, g, \mu), l')$ is defined as $\mathsf{disc\_succ}(t, (l, \zeta)) \triangleq (l', \zeta')$ where $\mathbf{v}' \in \zeta'$ iff $\exists \mathbf{v} \in \zeta \cap \mathsf{tpc}(l) \cap g.(\mathbf{v}, \mathbf{v}') \in \mu \wedge \mathbf{v}' \in \mathsf{tpc}(l')$.

Given disjoint actions A_i, an interaction is a subset of actions $\alpha \subseteq \bigcup_i A_i$ containing at most one action per component. Given a set of interactions $\gamma \subseteq 2^{\bigcup_i A_i}$, $Act(\gamma)$ denotes the actions in γ, that is, $Act(\gamma) \triangleq \bigcup_{\alpha \in \gamma} \alpha$. A PTS $\|_\gamma C_i$ is the composition of components C_i for the interaction set γ such that $Act(\gamma) \triangleq \bigcup_i A_i$. For n components $C_i \triangleq (L_i, l_0^i, \mathcal{X}_i, \mathcal{P}_i, A_i, T_i, \mathsf{tpc}_i, \mathcal{D}_i)$ with $L_i \cap L_j = \emptyset$, $A_i \cap A_j = \emptyset$, $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$, for any $i \neq j$, the *composition* $\|_\gamma C_i$ with respect to γ is defined by $(L, \overline{l}_0, \mathcal{X}, \mathcal{P}, \gamma, T_\gamma, \mathsf{tpc})$ where $\overline{l}_0$ is $(l_0^1, \ldots, l_0^n)$, $\mathcal{X}, \mathcal{P}, L, \mathsf{tpc}(\overline{l})$ are respectively $\bigcup_i \mathcal{X}_i$, $\bigcup_i \mathcal{P}_i$, $\times_i L_i$, $\bigcap_i \mathsf{tpc}_i(l_i)$, and T_γ is such that for $\alpha \triangleq \{a_i\}_{i \in I}$, $\overline{l} \xrightarrow{\alpha,g,\mu} \overline{l}'$ where $\overline{l}$ is $(l_1, \ldots, l_n)$, $g \triangleq \bigcap_{i \in I} g_i$, $\mu \triangleq \bigcap_{i \in I} \mu_i$, and $\overline{l}'(i)$ is l_i if $(i \notin I)$ else l_i' for $l_i \xrightarrow{a_i,g_i,\mu_i} l_i'$.

Figure 1 illustrates a system of 2 PTAs C_0, C_1 interacting on $\{c_0, c_1\}$. Initially, both C_0 and C_1 execute locally a, resp. b in either way. This is followed by a synchronisation on c_0 and c_1. We note that because y is reset on each transition, $y \leq x$ is a global property of the system. We also note that if $q = 7$ the system is deadlocked for any value of r: we have that $r \leq 3$ by the invariant of l_{10}, and because C_1 cannot stay in l_{11} for more than 3 units of time, it is impossible for the action a to be executed. This is a simple illustration showing that local parameter bounds might need to be tightened in the composed system in order to not introduce deadlocks.

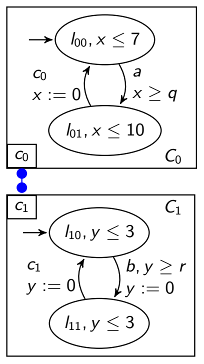


Fig. 1. A PTS

Definition 2 (Parameter Synthesis Problem). *Given a system* $\mathcal{S}$*, parameter constraints*[3] ϕ_p*, and a safety property* ϕ_{safe}*, a parameter synthesis problem is to find an assignment* $\mathbf{v}$ *for* $\mathcal{P}$ *such that* $\mathbf{v}$ *satisfies* ϕ_p *and* $\mathcal{S}(\mathbf{v})$ *satisfies* ϕ_{safe}*,* $\mathcal{S}(\mathbf{v}) \models \phi_{safe}$*. A satisfying assignment* $\mathbf{v}$ *is called a* solution.

[3] Parameter constraints are conjunctions of inequalities on $\mathcal{P}$ and $\mathbb{R}$ such as $q \in [0, 6]$.

3 Compositional Parameter Synthesis

We show how the method from [4] can be adapted to compute $\psi_{\mathcal{S}}$ in Formula (1). There are three steps to generate: (1) interaction invariants from γ; (2) component invariants from components with history clocks; (3) relations on history clocks.

Interaction Invariants. Interaction invariants are over-approximations of global locations. As their computation depends only on γ, it does not change in the parametric setup. Consequently, we omit its definition (to be found in [22, 5]) and instead illustrate it by means of our running example. We recall that $\gamma \triangleq \{a, b, \{c_0, c_1\}\}$. If a happens from l_{00} and l_{10}, C_0 reaches l_{01} while C_1 remains in l_{10}. If $\{c_0, c_1\}$ happens from l_{01} and l_{11}, l_{00} and l_{10} are reached. Continuing this reasoning for all combinations, we obtain as interaction invariant $\mathcal{I}(\gamma)$ the formula $(l_{00} \wedge l_{10}) \vee (l_{00} \wedge l_{11}) \vee (l_{01} \wedge l_{10}) \vee (l_{01} \wedge l_{11})$.

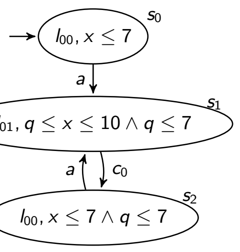
Component Invariants. Component invariants characterise the reachable states of components when considered alone. Given a component C with locations L, we assume that the symbolic states resulting from the computation of its parametric zone graph are $\{s_i\}_I$ with s_i being (l_j, ζ_j). We consider the following formula:

$$\mathcal{I}(C) \triangleq \bigwedge_{i \in I} (s_i \rightarrow (l_j \wedge \zeta_j)) \wedge \bigwedge_{l \in L} (l \rightarrow \bigvee_{l \in s} s). \tag{2}$$

By abuse of notation, l_j is used to denote the predicate that holds whenever C is at location l_j and $l \in s$ holds if $s = (l, \zeta)$.

The parametric zone graph of C_0 for our running example (Figure 1), as computed with Imitator [2], is in Figure 2. By Equation (2), $\mathcal{I}(C_0)$ is as follows:

$$\begin{aligned} \mathcal{I}(C_0) = & s_0 \rightarrow (l_{00} \wedge x \leq 7) \wedge \\ & s_1 \rightarrow (l_{01} \wedge q \leq x \leq 10 \wedge q \leq 7) \wedge \\ & s_2 \rightarrow (l_{00} \wedge x \leq 7 \wedge q \leq 7) \wedge \\ & l_{00} \rightarrow (s_0 \vee s_2) \ \wedge \ l_{01} \rightarrow s_1 \end{aligned}$$



s_0
$l_{00}, x \leq 7$
a
s_1
$l_{01}, q \leq x \leq 10 \wedge q \leq 7$
a
c_0
s_2
$l_{00}, x \leq 7 \wedge q \leq 7$

Fig. 2.

The formula in Equation (2) is more precise than the one in [4]. There, the choice was to take the disjunction of $l_j \wedge \zeta_j$ as an invariant. In a parametric setup, such an encoding is not enough: since $s_0 \vee s_2$ reduces to $l_{00} \wedge x \leq 7$, the relation $q \geq 7$ is lost.

More importantly, the formula in Equation (2) is not necessarily an invariant. For instance, for the valuation $\mathbf{v} \triangleq \{q = 8\}$, $CI(C_0)(\mathbf{v})$ reduces to false. $\mathcal{I}(C)$ is an invariant only under the parameter valuations which satisfy the parameter constraints in it. Let us denote by $K_p(C)$ the parameter constraints in Equation (2). $K_p(C)$ is obtained from $\mathcal{I}(C)$ by a similar approach as in [2], that is, by seeing clocks as existential variables and doing quantifier elimination. For instance, $K_p(C_0)$ is $q \leq 7$.

Proposition 1 *For a component C with parameter constraints K_p, $\mathcal{I}(C)(\mathbf{v})$ is an invariant of $C(\mathbf{v})$ for any $\mathbf{v}$ such that $\mathbf{v} \models K_p$.*

History Clocks & Auxiliary Constraints In general, component and the interaction invariants are not enough to prove global properties, especially when such properties involve relations between clocks in different components. In the case of $\exists\forall$ solving, a weak invariant leads to no solution: there is not enough information to synthesise parameters such that the global property holds. For instance, in our toy example, we cannot find parameters such that $y \leq x$ holds by only having at hand the invariants for components and interactions: there are no relations relating both

x and y. By means of *history clocks* we are able to derive new global constraints from the simultaneity of interactions and the synchrony of time progress. These new constraints make it possible to successfully find parameter valuations such that global properties hold.

Adding History Clocks. History clocks are associated with actions and interactions. For a component C we use C^h to denote its extension with history clocks. The extension of the system is obtained from the extensions of the components alone together with the history clocks for interactions. As an illustration, Figure 3 shows the extension of the system in Figure 1.

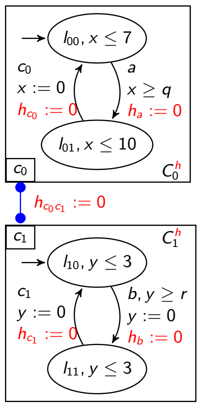


Fig. 3.

The intuition behind history clocks is as follows. When interaction α takes place, the history clocks h_α and h_a associated to α and to any action $a \in \alpha$ are reset. Thus they measure the time passed from the last occurrence of α, respectively of a. Since there is no timing constraint involving history clocks, the behaviour of the components is not changed by the addition of history clocks.

For timed automata, the zone graph is finite, consequently so is the computation of $\mathcal{I}(C)$ and $\mathcal{I}(C^h)$ as in Equation (2). This is no longer the case in the parametric setup. For instance, the parametric zone graph of the component C in Figure 4 has two symbolic states $l_0 \wedge x \leq 5$ and $l_1 \wedge x \leq 3 \wedge r \leq 5$ while the one of C^h contains infinitely many symbolic states such as $l_1 \wedge x \leq 3 \wedge r \leq 5 \wedge y = h_c \geq h_a \wedge y + 3k \geq h_a$ for $k \in \mathbf{N}$. We note that, though one could find particular solutions depending on the systems in cause, since the reachability problem is undecidable for parametric timed automata [1], one cannot hope for general solutions.

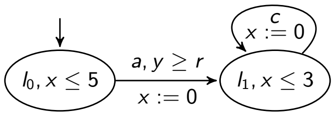


Fig. 4.

Generating Interaction Equalities from History Clocks. The basic underlying observation is that a history clock h_a for an action a from a last executed interaction α is necessarily *less* than any h_β with β another interaction containing a. This is because the clocks of the actions in α are the last ones being reset. Consequently, given a common action a of $\alpha_0, \alpha_2, \ldots, \alpha_p$, h_a is the minimum of h_{α_i}, $h_a \triangleq \min_{0 \leq i \leq p} h_{\alpha_i}$. The invariant for a given interaction set γ is denoted as $\epsilon(\gamma)$ and defined as follows:

$$\epsilon(\gamma) \triangleq \bigwedge_{a \in Act(\gamma)} h_a = \min_{\alpha \in \gamma, a \in \alpha} h_\alpha.$$

For our running example, $\epsilon(\{c_0, c_1\})$ is simply $h_{c_0} = h_{c_1}$.

Generating Inequalities from Conflicting Interactions. Without conflicts, that is, when interactions do not share any action, $\epsilon(\gamma)$ is quite tight in the sense that it is essentially a conjunction of equalities. However, $\epsilon(\gamma)$ is weaker in the presence of conflicts because any action in conflict can be used in different interactions. The disjunctions (implicit in the definition of $\min$) in $\epsilon(\gamma)$ reflect precisely this uncertainty. History clocks on interactions are introduced to capture the time lapses between conflicting interactions. The basic information exploited in [4] is that when two conflicting interactions compete for the same action a, no matter which one is first, the other one must wait until the component which owns a is again able to execute a. This has been referred to as a "separation constraint" for conflicting interactions and was defined as the following invariant:

$$\sigma(\gamma) \triangleq \bigwedge_{a \in Act(\gamma)} \bigwedge_{\substack{\alpha \neq \beta \in \gamma \\ a \in \alpha \cap \beta}} |h_\alpha - h_\beta| \geq k_a$$

where $|x|$ denotes the absolute value of x and k_a represents the minimum elapsed time between two consecutive executions of a. In the case of timed automata, the computation of such minimum elapses follows the classical [11] which consists in finding a shortest path in a weighted graph built from the zone graph associated to a timed automaton. The extension to PTAs follows the same construction.

A more practical solution is to construct an observer. To compute the delay between two consecutive executions of a in C, we can check if $C \|_a \mathcal{O}^a \models \psi^a_{obs}$ is not true, where $\mathcal{O}^a$ is the automaton in Figure 5 and ψ^a_{obs} is $\Box \neg l_2$.

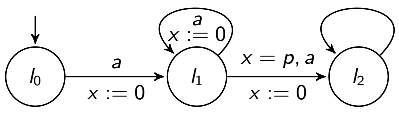


Fig. 5. An observer for computing k_a

In our running example, there are no conflicting interactions. If there were another component with action c_2 interacting with C_0 by means of interaction $\{c_0, c_2\}$ then $\{c_0, c_2\}$ is in conflict with $\{c_0, c_1\}$. The separation between them is given by the time elapse between two consecutive c_0 which in this particular case is simply q.

The formulae computed throughout this section are invariants. Together, they form the over-approximation $\psi_{\mathcal{S}}$ in the $\exists\forall$ Formula from (1).

Proposition 2 *Given $\mathcal{S} \triangleq \|_\gamma C_i$, let $K_p(C_i)$ be the parameter constraints for C_i and let $\psi_{\mathcal{S}}$ denote the formula $\bigwedge_i \mathcal{I}(C_i^h) \wedge \mathcal{I}(\gamma) \wedge \epsilon(\gamma) \wedge \sigma(\gamma)$ after the elimination of history clocks. We have that for any $\mathbf{v}$ such that $\mathbf{v} \models \wedge_i K_p(C_i)$, $\psi_{\mathcal{S}}(\mathbf{v})$ is an invariant of $\mathcal{S}$.*

Finding Satisfying Instances for $\exists\forall$ Formulae. We recall that we reduce our synthesis problem to solving the $\exists\forall$ formulae in (1). For illustration, we show how the formula looks like for our running example. We have the following formulae:

$$\begin{aligned}
\mathcal{I}(C_0^h) = & s_0 \rightarrow (l_{00} \wedge x \leq 7 \wedge x = h_a = h_{c_0}) \wedge \\
& s_1 \rightarrow (l_{01} \wedge q + h_a \leq x \leq min(10, h_a + 7) \wedge x = h_{c_0}) \wedge \\
& s_2 \rightarrow (l_{00} \wedge x \leq 7 \wedge q + h_a - 10 \leq x \leq min(7, h_a) \wedge x = h_{c_0}) \wedge \\
& l_{00} \rightarrow (s_0 \vee s_2) \ \wedge \ l_{01} \rightarrow s_1 \\
\mathcal{I}(C_1^h) = & s_0' \rightarrow (l_{10} \wedge y \leq 3 \wedge y = h_b = h_{c_1}) \wedge \\
& s_1' \rightarrow (l_{11} \wedge r + y \leq h_{c_1} \leq y + 3 \wedge y = h_b) \wedge \\
& s_2' \rightarrow (l_{10} \wedge y \leq 3 \wedge y \leq h_b \leq y + 3 \wedge y = h_{c_1}) \wedge \\
& l_{10} \rightarrow (s_0' \vee s_2') \ \wedge \ l_{11} \rightarrow s_1'
\end{aligned}$$

By inspecting $\mathcal{I}(C_0^h)$ and $\mathcal{I}(C_1^h)$, we can derive that $K_p(C_0^h)$ is $q \leq 7$ and that $K_p(C_1^h)$ is $r \leq 3$. Assuming ϕ_{safe} is $y \leq x$, the $\exists\forall$ Formula (1) is:

$$\exists q, r. q \leq 7 \wedge r \leq 3. \forall l \in L, s \in S, x, y. \mathcal{I}(\gamma) \wedge qe\big(\mathcal{I}(C_0^h) \wedge \mathcal{I}(C_1^h) \wedge h_{c_0} = h_{c_1}\big) \rightarrow y \leq x$$

where L denotes $\{l_{00}, l_{01}, l_{10}, l_{11}\}$, S denotes $\{s_0, s_1, s_2, s_0', s_1', s_2'\}$, and qe denotes the result of eliminating the history clocks. Since both $x = h_{c_0}$ and $y \leq h_{c_1}$ are invariants, together with $h_{c_0} = h_{c_1}$, it can be derived that $y \leq x$. Consequently, for any $r \leq 3$ and $q \leq 7$, the system satisfies $y \leq x$.

We also note that for an interaction property expressing that a happens before b to hold, q must be smaller or equal than 3, though the upper bound in the local constraint is 7. This is because s_1 must be reached while still at s_0' where $y = h_{c_1} \leq 3$. Since $q \leq x$ and $x = h_{c_0}$ we have that $q \leq 3$ by using $h_{c_0} = h_{c_1}$. This shows that history clocks forbid parameter valuations which satisfy local parameter constraints but which could introduce deadlocks in the system.

When the ∃∀SMT solver returns `unsat`, there are two interpretations: (1) either ϕ_{safe} does not hold or (2) $\psi_{\mathcal{S}}$ is too coarse. To check (1), one might apply the method from [4] and feed $\psi_{\mathcal{S}} \wedge \phi_{safe}$ to an SMT solver. If the result is `unsat`, then ϕ_{safe} is not a property of the system. For instance, if for our running example we take $x < y$ as ϕ_{safe}, the ∃∀SMT solver returns `unsat` for Formula (1). Since $\psi_{\mathcal{S}} \wedge x < y$ returns `unsat` as well, we know that $x < y$ is not valid.

We conclude the section by stating the correctness of our approach which follows from the fact that $\psi_{\mathcal{S}}(\mathbf{v})$ is an over-approximation (Proposition 2).

Proposition 3 *Given $\mathcal{S} \triangleq \|_{\gamma} C_i$ let $K_p(C_i)$ denote the parameter constraints for C_i. If $\mathbf{v}$ is such that it satisfies Formula (1) together with $\bigwedge_i Kp(C_i)$ then $\mathbf{v}$ is a solution to the parameter synthesis problem, i.e., $\mathbf{v} \in \phi_p$ and $\mathcal{S}(\mathbf{v}) \models \phi_{safe}$.*

4 Experiments and Extensions

We have implemented a prototype[4] to experiment with our approach. The prototype takes as input components as PTAs in Imitator [2], a file describing the interactions, the constraints over parameters and a safety property. It uses EFSMT [9] and Z3 [23] to return either `unsat` or a parameter assignment under which the safety property holds. We have also connected our prototype with the one in [4] such that, in case the result is `unsat`, we check if the global property given as input is not actually false. With respect to performance, our prototype returns an answer within a second for variations on toy benchmarks such as the train gate controller or the temperature controller with as many as 16 trains, respectively rods. As final notes, we make two observations: (1) our experiments with invariants without history clocks show that these invariants are clearly weaker in the sense that the solver does not find any parameter valuations; (2) on the negative side, even for minimal models of production lines with filling and packaging machines, the computation of the set of reachable states does not terminate.

Due to our encoding of the parameter synthesis problem as ∃∀SMT formulae, we can readily solve the following extensions of parameter synthesis for PTS.

Beyond PTA. Using the expressiveness of decidable ∃∀-constraints, one can encode guards such as $t_1 + 3\,t_4 \geq 10$ and also non-linear arithmetic constraints, as obtained from some richer classes of hybrid automata.

Quantitative Synthesis. Our method does not generate optimised paramater values, since this would require an additional quantifier alternation [9]. However, EFSMT can be modified for optimisation by using a MaxSMT solver (e.g. νZ [6]) instead of an SMT solver for formulae of existential polarity (the so-called E-solver).

Robustness Synthesis. The imprecision of systems may be modelled by means of universally quantified, bounded variables. For example, one may model the imprecision for a guard $t_1 > 2$ by $t_1 > 2 + \delta$, for $\delta \in [-0.05, 0.05]$ by simply adding $\forall \delta \in [-0.05, 0.05]$ in the ∃∀SMT formula.

Interaction Properties in LTL. Interaction properties such as "eventually interaction a will happen before b", can effectively be transformed into safety properties based on the encodings of a corresponding Büchi automata along the lines proposed for bounded synthesis [16] or for bounded model checking [24].

[4] The source code and examples can be found at github.com/astefano/efsmt_coverts.

Acknowlegdement. We warmly thank Étienne André for suggesting us the construction of the observer to compute the separations in Section 3.

References

1. R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *ACM*, pages 592–601, 1993.
2. É. André. IMITATOR II: A tool for solving the good parameters problem in timed automata. In *INFINITY*, 2010.
3. É. André and R. Soulat. Synthesis of timing parameters satisfying safety properties. In *Reachability Problems*, 2011.
4. L. Aştefănoaei, S. B. Rayana, S. Bensalem, M. Bozga, and J. Combaz. Compositional invariant generation for timed systems. In *TACAS*, 2014.
5. S. Bensalem, M. Bozga, J. Sifakis, and T.-H. Nguyen. Compositional verification for component-based systems and application. In *ATVA*, 2008.
6. N. Bjørner, A. Phan, and L. Fleckenstein. νz - an optimizing SMT solver. In *TACAS*, LNCS, pages 194–199. Springer, 2015.
7. R. Bruttomesso, A. Carioni, S. Ghilardi, and S. Ranise. Automated analysis of parametric timing-based mutual exclusion algorithms. In *NFM*, 2012.
8. C. Cheng, T. Guelfirat, C. Messinger, J. O. Schmitt, M. Schnelte, and P. Weber. Semantic degrees for Industrie 4.0. *CoRR*, abs/1505.05625, 2015.
9. C. Cheng, N. Shankar, H. Ruess, and S. Bensalem. EFSMT: A logical framework for cyber-physical systems. *CoRR*, abs/1306.3456, 2013.
10. A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. Parameter synthesis with IC3. In *FMCAD*, pages 165–168. IEEE, 2013.
11. C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1992.
12. W. Damm, C. Ihlemann, and V. Sofronie-Stokkermans. Ptime parametric verification of safety properties for reasonable linear hybrid automata. *Mathematics in Computer Science*, 5(4), 2011.
13. T. Dang, T. Dreossi, and C. Piazza. Parameter synthesis through temporal logic specifications. In *FM*, LNCS, pages 213–230. Springer, 2015.
14. A. Donzé. Breach, A toolbox for verification and parameter synthesis of hybrid systems. In *CAV*, 2010.
15. J. Faber, C. Ihlemann, S. Jacobs, and V. Sofronie-Stokkermans. Automatic verification of parametric specifications with complex topologies. In *IFM*, 2010.
16. B. Finkbeiner and S. Schewe. Bounded synthesis. *STTT*, 15(5-6):519–539, 2013.
17. G. Frehse, S. K. Jha, and B. H. Krogh. A counterexample-guided approach to parameter synthesis for linear hybrid automata. In *HSCC*, LNCS, pages 187–200. Springer, 2008.
18. L. Fribourg and U. Kühne. Parametric verification and test coverage for hybrid automata using the inverse method. *Int. J. Found. Comput. Sci.*, 24, 2013.
19. T. A. Henzinger and H. Wong-Toi. Using HyTech to synthesize control parameters for a steam boiler. In *FMIA*, 1995.
20. T. Hune, J. Romijn, M. Stoelinga, and F. W. Vaandrager. Linear parametric model checking of timed automata. *J. Log. Algebr. Program.*, 52-53, 2002.
21. A. Jovanovic, D. Lime, and O. H. Roux. Integer parameter synthesis for timed automata. In *TACAS*, 2013.
22. A. Legay, S. Bensalem, B. Boyer, and M. Bozga. Incremental generation of linear invariants for component-based systems. In *ACSD*, 2013.
23. L. Moura and N. Bjørner. Efficient e-matching for smt solvers. In *Proceedings of CADE*, 2007.
24. L. Moura, H. Rueß, and M. Sorea. Lazy theorem proving for bounded model checking over infinite domains. In *CADE*, LNCS, pages 438–455. Springer, 2002.

25. V. Sofronie-Stokkermans. Hierarchical reasoning for the verification of parametric systems. In *IJCAR*, 2010.
26. F. Wang. Symbolic parametric safety analysis of linear hybrid systems with bdd-like data-structures. In *CAV*, LNCS, pages 295–307. Springer, 2004.