Parameter Synthesis with IC3

Alessandro Cimatti
Fondazione Bruno Kessler
Email: cimatti@fbk.eu

Alberto Griggio
Fondazione Bruno Kessler
Email: griggio@fbk.eu

Sergio Mover
Fondazione Bruno Kessler
Email: mover@fbk.eu

Stefano Tonetta
Fondazione Bruno Kessler
Email: tonettas@fbk.eu

***Abstract*—Parametric systems arise in different application domains, such as software, cyber-physical systems or tasks scheduling. A key challenge is to estimate the values of parameters that guarantee the desired behaviours of the system.**

In this paper, we propose a novel approach based on an extension of the IC3 algorithm for infinite-state transition systems. The algorithm finds the feasible region of parameters by complement, incrementally finding and blocking sets of "bad" parameters which lead to system failures. If the algorithm terminates we obtain the precise region of feasible parameters of the system.

We describe an implementation for symbolic transition systems with linear constraints and perform an experimental evaluation on benchmarks taken from the domain of hybrid systems. The results demonstrate the potential of the approach.

I. Introduction

Parametric systems arise in many application domains from real-time systems to software to cyber-physical systems. In these applications, the system is often part of a larger environment, and the designer has to define the system relative to some unknown parameters of the environment. The design of a robust system requires the verification to not rely on concrete values for the parameters but to prove the correctness of the system for a certain region of values. The use of parameters is fundamental in the early phases of the development, giving the possibility to explore different design choices. In fact, a parametric system represents a set of (non-parametric) systems, one for each valuation of the parameters.

A key challenge for the design of parametric systems is the estimation of the parameter valuations that guarantee the correct behavior of the system. A manual estimation of these values is time consuming and does not allow to find optimal solutions for specific design problems. Therefore, a fundamental problem is to automatically synthesize the maximal region of parameter valuations for which the system satisfies some properties.

In this paper, we focus on SMT-based verification of invariant properties and how to extend SMT-based algorithms to solve the synthesis problem. The general approach works by complement, iteratively building the set of "bad" parameter valuations. It relies on the enumeration of counterexamples violating the properties, extracting from the counterexample a region of "bad" parameter valuations by quantification of the state variables.

The novel contribution of this paper is a new synthesis algorithm based on IC3, one of the major recent breakthroughs in SAT-based model checking, and lately extended to the SMT case. The key idea of the synthesis algorithm is to exploit the features of IC3. First, IC3 finds counterexamples consisting of a sequence of states $s_0, s_i, \ldots, s_k$ that are guaranteed to reach a bad state in $k-i$ steps; this is exploited to quantify the state-variables to find a region of bad parameter violations, without the need of quantify over full paths. Second, the internal structures of IC3 allows our extension to be integrated in a fully incremental fashion, reusing all the previously discovered information.

Various approaches already solve the parameter synthesis problem for several kind of systems, like infinite-state transition systems [5], timed and hybrid automata [10], [13], [9], [8], [2]. The advantages of the new algorithm with respect to other approaches are that it synthesizes an optimal region of parameters (unlike [9], [2]), it is incremental and applies quantifier elimination only to small formulas (unlike [9], [8]), and it avoids computing the whole set of the reachable states (unlike [10], [13]).

We implemented the algorithm for symbolic transition systems with linear constraints and performed an experimental evaluation on benchmarks on real-time and hybrid systems. We compared the approach with similar SMT-based techniques and with other techniques based on the computation of the reachable states. The results show the potential of the approach.

II. Background

A. Transition Systems

A *transition system* S is a tuple $S = \langle X, I, T \rangle$ where X is a set of (state) variables, $I(X)$ is a formula representing the initial states, and $T(X, X')$ is a formula representing the transitions. In this paper, we shall deal with *linear rational arithmetic* formulas, that is, Boolean combinations of propositional variables and linear inequalities over rational variables. A *state* of S is an assignment to the variables X. A *path* of S is a finite sequence $s_0, s_1, \ldots, s_k$ of states such that $s_0 \models I$ and for all i, $0 \leq i < k$, $s_i, s'_{i+1} \models T$. Given a formula $P(X)$, the *verification problem* denoted with $S \models P$ is the problem to check if for all paths $s_0, s_1, \ldots, s_k$ of S, for all i, $0 \leq i \leq k$, $s_i \models P$. The dual problem is the *reachability problem*, which is the problem to find a path $s_0, s_1, \ldots, s_k$ of S such that $s_k \models \neg P$. $P(X)$ represents the "good" states, while $\neg P$ represents the "bad" states.

B. Parameter Synthesis

In parametric systems, besides the standard constants, the formulas can include also *parameters*, which are rigid symbols with "unknown" values.

Let U be the set of parameters. A *parameter valuation* is as assignment to the parameters. Given a formula ϕ and a parameter valuation γ, we denote with $\gamma(\phi)$ the formula obtained from ϕ by replacing each parameter in U with the assignment given by γ.

A *parametric transition system* S is a tuple $S = \langle U, X, I, T\rangle$ where U is the set of parameters, X is the set of variables, $I(U, X)$ is the initial formula, and $T(U, X, X')$ is the transition formula. Each parameter valuation γ induces a transition system $S_\gamma = \langle X, \gamma(I), \gamma(T)\rangle$.

Given a parametric transition system $S = \langle U, X, I, T\rangle$ and a formula $P(U, X)$, we say that a parameter valuation γ is *feasible* iff $S_\gamma \models \gamma(P)$. The *parameter synthesis problem* is the problem of finding a set $\rho(U)$ of parameter valuations such that, for every $\gamma \in \rho$, $S_\gamma \models \gamma(P)$ (i.e., a set of feasible parameter valuations). We say that $\rho(U)$ is *optimal* if it contains all the feasible parameter valuations.

C. IC3 *with SMT*

IC3 is an efficient algorithm for the verification of finite-state systems, with Boolean state variables and propositional logic formulas, introduced by Bradley in [4]. IC3 was subsequently extended to the SMT case in [6], [11]. In the following, we present its main ideas, following the description of [6]. For brevity, we have to omit several important details, for which we refer to [4], [6], [11].

Let S and P be a transition system and a set of good states as in §II-A. The IC3 algorithm tries to prove that $S \models P$ by finding a formula $F(X)$ such that: (i) $I(X) \models F(X)$; (ii) $F(X) \wedge T(X, X') \models F(X')$; and (iii) $F(X) \models P(X)$.

In order to construct an inductive invariant F, IC3 maintains a sequence of formulas (called *trace*) $F_0(X), \dots, F_k(X)$ such that: (i) $F_0 = I$; (ii) $F_i \models F_{i+1}$; (iii) $F_i(X) \wedge T(X, X') \models F_{i+1}(X')$; (iv) for all $i < k$, $F_i \models P$;

The algorithm proceeds incrementally, by alternating two phases: a blocking phase, and a propagation phase. In the *blocking* phase, the trace is analyzed to prove that no intersection between F_k and $\neg P(X)$ is possible. If such intersection cannot be disproved on the current trace, the property is violated and a counterexample can be reconstructed. During the blocking phase, the trace is enriched with additional formulas, that can be seen as strengthening the approximation of the reachable state space. At the end of the blocking phase, if no violation is found, $F_k \models P$.

The *propagation* phase tries to extend the trace with a new formula F_{k+1}, moving forward the clauses from preceding F_i's. If, during this process, two consecutive elements of the trace (called *frames*) become identical (i.e. $F_i = F_{i+1}$), then a fixpoint is reached, and IC3 can terminate with F_i being an inductive invariant proving the property.

More in detail, in the blocking phase, IC3 maintains a set of pairs (s, i), where s is a set of states that can lead to a bad state, and $i > 0$ is a position in the current trace. New formulas (in the form of clauses) to be added to the current trace are derived by (recursively) proving that a set s of a pair (s, i) is unreachable starting from the formula F_{i-1}. This is done by checking the satisfiability of the formula:

$$F_{i-1} \wedge \neg s \wedge T \wedge s'. \qquad (1)$$

If (1) is unsatisfiable, and s does not intersect the initial states I of the system, then $\neg s$ is *inductive relative to* F_{i-1}, and IC3 strengthens F_i by adding $\neg s$ to it[1], thus *blocking* the bad state s at i. If, instead, (1) is satisfiable, then the overapproximation F_{i-1} is not strong enough to show that s is unreachable. In this case, let p be a subset of the states in $F_{i-1} \wedge \neg s$ such that all the states in p lead to a state in s' in one transition step. Then, IC3 continues by trying to show that p is not reachable in one step from F_{i-2} (that is, it tries to block the pair $(p, i-1)$). This procedure continues recursively, possibly generating other pairs to block at earlier points in the trace, until either IC3 generates a pair $(q, 0)$, meaning that the system does not satisfy the property, or the trace is eventually strengthened so that the original pair (s, i) can be blocked.

A key difference between the original Boolean IC3 and its SMT extensions in [6], [11] is in the way sets of states to be blocked or generalized are constructed. In the blocking phase, when trying to block a pair (s, i), if the formula (1) is satisfiable, then a new pair $(p, i-1)$ has to be generated such that p is a cube in the *preimage of s wrt. T*. In the propositional case, p can be obtained from the model μ of (1) generated by the SAT solver, by simply dropping the primed variables occurring in μ. This cannot be done in general in the first-order case, where the relationship between the current state variables X and their primed version X' is encoded in the theory atoms, which in general cannot be partitioned into a primed and an unprimed set. The solution proposed in [6] is to compute p by existentially quantifying (1) and then applying an *under-approximated* existential elimination algorithm for linear rational arithmetic formulas. Similarly, in [11] a theory-aware generalization algorithm for linear rational arithmetic (based on interpolation) was proposed, in order to strengthen $\neg s$ before adding it to F_i after having successfully blocked it.

III. Parameter Synthesis with IC3

A. Solving the synthesis problem with reachability

A naive approach to synthetize the set of parameters $\rho(U)$ is to incrementally find the complement set $\beta(U)$ (thus, $\rho = \neg\beta$) of unfeasible parameter valuations rephrasing the problem as a reachability problem for a transition system S_ρ and iteratively removing the counterexamples to $S_\rho \models P$.

More specifically, given the parametric transition system $S = \langle U, X, I, T\rangle$, the algorithm keeps an over-approximation $\rho(U)$ (initially true) of the safe region. The encoding of S is the transition system $S_\rho = \langle X \cup P, I_\rho, T_\rho\rangle$ where $T_\rho = T \wedge \bigwedge_{p \in U} p' = p$ forces parameters to not change their value in the evolution of the system and $I_\rho = I \wedge \rho$ restricts the parameter valuations to the over-approximation.

[1]In fact, $\neg s$ is actually *generalized* before being added to F_i. Although this is quite important for the effectiveness of IC3, here for simplicity we shall not discuss this.

At every iteration, a new parameter valuation is removed from ρ. The algorithm terminates if it proves that $S_\rho \models P$, and ρ is the solution to the synthesis problem.

This simple approach does not work in the context of infinite-state transition systems, where in general the possible number of counterexamples and the values of the parameters are infinite. For this reason, we need an algorithm that removes a set of parameters, instead of a single point.

B. *Description of the synthesis algorithm with* IC3

We embed a reasoning similar to the naive algorithm in IC3, exploiting the generalization of counterexamples and the incremental behaviour. The generalization avoids the explicit enumeration of the counterexamples, while the incrementality allows us to *completely* reuse all the clauses learned by IC3 across different safety checks.

Therefore, IC3 is used to prove that $S_\rho \models P$. If it is successful (recall that in the SMT extension, the problem is undecidable), we can conclude that ρ is a set of feasible parameters and, in particular, is optimal. Instead, if there exists a set of parameters such that $S \not\models P$, IC3 will might find a counterexample to P. The counterexample is found in the blocking phase as a sequence $\pi := (s_0, 0), \ldots, (s_n, n)$, where $s_0 \models I'$, $s_n \models \neg P$ and for $0 < i < n-1$, $s_i \wedge T' \models s_{i+1}$. Possibly, π does not represent a single path of the system that reaches a violation, but a set of paths that reach $\neg P$. This is an intrisic feature of IC3, which generalizes the counterexamples to induction found in the blocking phase, trying to block set of states rater than a single state. The state s_o represents a set of states that will eventually reach $\neg P$. Thus, we compute from s_0 a set of bad parameters $\beta_{s_o}(U)$ that will eventually reach s_n: $\beta_{s_o}(U) := \exists X.s_o(U, X)$. We rely on a quantifier elimination procedure to obtain a quantifier-free formula for β_{s_o}.

The algorithm refines its conjecture about the unfeasible parameters of the system. Let $\beta' := \beta \vee \beta_{s_o}$ and $\rho' := \rho \wedge \neg\beta_{s_o}$ be the new approximations of unfeasible and feasible regions of parameters. We have to prove that $S_{\rho'} \models P$. We perform the verification incrementally, reusing all the frames of IC3. Since $\rho' := \rho \wedge \neg\beta_{s_o}$, we have that $S_{\rho'} = \langle X \cup P, I_\rho \wedge \neg\beta_{s_o}, T_\rho \rangle$.[2] Thus, we incrementally encode $S_{\rho'}$ strengthening the initial condition and the transition relation used in the algorithm, and also strengthening the first frame kept by the IC3 algorithm (i.e. $F_0 := F_0 \wedge \neg(\beta_{s_o})$). The strengthening of F_0 removes the state s_o from I (possibly blocking also other bad states).

Since S_ρ is an overapproximation of $S_{\rho'}$, the invariant mantained by IC3 (i.e. $F_0 = I$, $F_i \models F_{i+1}$, $F_i \models P$ and $F_i(X) \wedge T(X, X') \models F_{i+1}(X')$) also holds for the new problem $S_{\rho'} \models P$.

From this point, we rely on the usual behaviour of IC3, which tries to block $(s_1, 1)$ with the strengthened frame F_0. The algorithm terminates if either P is proved or the F_0 becomes unsatisfiable, showing that ρ is empty.

[2]We also add $\neg\beta_{s_o}$ also to T_ρ, since it is an inductive invariant of $S_{\rho'}$.

```
bool ParamIC3 ($U$, $I$, $T$, $P$):
  $\beta(U)$ = $\bot$  # underapproximation of the unfeasible parameters
  trace = [$I$]  # first elem of trace is init formula
  trace.push()  # add a new frame to the trace
  while True:
      # blocking phase
      while there exists a cube $c$ s.t. trace.last() $\wedge T \wedge c$ is satisfiable
              and $c \models \neg P$:
          recursively block the pair $(c, \text{trace.size()} - 1)$
          if a pair $(p, 0)$ is generated:
              $\beta_p$ = $\exists X.p(U, X)$
              $\beta := \beta \vee \beta_p$.
              $I := I \wedge \neg\beta_p$ and $T := T \wedge \neg\beta_p$ .
              add $\neg\beta_p$ to trace[0].
              remove $(p, 0)$ from the set of states to be blocked.
              if $I \models \bot$  # the initial states are empty
                  return $\bot$

      # propagation phase
      trace.push()
      for $i = 1$ to trace.size() $- 1$:
          for each clause $c \in$ trace[i]:
              if trace[i] $\wedge c \wedge T \wedge \neg c'$ is unsatisfiable:
                  add $c$ to trace[i+1]
          if trace[i] == trace[i+1]:
              return $\neg\beta$  # P proved, return good params region
```

Fig. 1. High-level description of ParamIC3. The bold and red text shows the code which differ from the IC3 algorithm used for verification.

We show the parameter synthesis algorithm ParamIC3 in the Figure 1, highlighting in red the modifications to the original IC3.

Theorem 1: Given a parametric transition system $S = \langle U, X, I, T \rangle$ and a formula $P(X)$, $\rho(U) := \text{ParamIC3}(U, I, T, P)$ is the *optimal set* of feasible parameter valuations.

C. *Optimizations*

We presented a version of the algorithm which computes a region of bad states β_{s_o} only from the initial states of $\pi := (s_0, 0), \ldots, (s_n, n)$. However, this is only one of the possible choices, since more general regions of bad parameters can be found considering each s_i in π. In fact, β_{s_o} is one of the extreme cases, while the other one is $\beta_n(U) := \exists X.(BMC_n)$, which encodes the set of all the parameters that may reach $\neg P$ in n steps, where BMC_n denotes $I^0 \wedge \bigwedge_{i=0}^{n-1} T^i \wedge \neg P^n$. However, the cost of eliminating the quantifiers grows as well, and it might in fact become impractical. In principle, one may consider the intermediate cases β_{s_i} (that is, the reachability of one of the intermediate states s_i in π) to trade the generality of the result with the cost of the quantifier elimination. Furthermore, we notice that for soundness we do not need the precise set β_{s_i}, but we can consider its under-approximations, since this still guarantees to remove only bad parameters valuations. As an advantage, in this case the quantifier elimination problems are easier to solve and are more general than β_{s_o}. In practice, we use an heuristic which tries to combine the precise and the under-approximated approach, enabling us to find a trade-off between generality and the cost of quantifier elimination. The heuristic that we use is described in the next Section.

IV. Related works

The IC3 [4] algorithm was first proposed to prove safety property for finite-state transition systems. Several approaches adapted the original algorithm to deal with infinite-state systems [6], [11], [12].

The works presented in [6], [11] may be used as backends to synthesize the parameters via reachability. However, they need to perform a quantifier elimination step on an entire path and they will not be able to exploit the information discovered by IC3 while finding violations to the property. Instead, [12] cannot be used as a reachability backends, since it is restricted to timed automata without parameters.

The parameter synthesis for infinite-state transition systems can be solved combining a reachability algorithm with a quantifier elimination procedure, as proposed by [8]. While the approach was proposed in the context of parametric timed automata, it may applied to infinite-state systems. Our approach follows the same general idea, which is iteratively find the unfeasible regions of parameters. However, a key difference is in the computation of a set of bad region with the quantifier elimination procedure, since we apply the quantifier elimination to a set of states and not to a set of traces. The approach proposed in [5] deals with infinite-state systems with an unbounded number of processes and several kind of properties, like mutual exclusion and deadlock detection. While this setting is more general than ours, it does not synthesize the entire region of parameters, but it instantiates the values of the parameters for a given template.

Other works [10], [13], [9], [8], [1], [2] synthesize parameters for real-time and hybrid systems. Tools like HyTech [10], TReX [3] or Red [13] synthesize the parameters computing the reachable states of the system. All these techniques are precise, but they are forced to compute the entire set of reachable states, which may be unfeasible in several cases. The technique proposed by Frehse [9] handles linear hybrid automata, using an approach similar to [8]. The approach is not precise and underapproximates the region of feasible parameters. Instead, we find the precise region of parameters. Finally, the tools IMITATOR [1] and HYMITATOR [2], which respectively handle timed and hybrid systems, solve a different but related problem to parameter synthesis. Given a parameter valuation, they compute all the parameter values that induce the same set of discrete traces as the given parameter valuation. This approach requires an initial assignments for the parameters and in general it will not find the maximum region of feasible parameters. We stress that our approach is not specific to timed and hybrid automata, but it may be applied to every infinite-state transition systems expressed using *Linear Rational Arithmetic*.

V. Experiments

We have implemented the algorithm described in the previous section on top of the fully symbolic SMT-based IC3 of [6]. The tool uses MathSAT [7] as backend SMT engine, and works on transition systems with linear arithmetic constraints.

Evaluation. Our evaluation consists of three parts. In the first, we compare our implementation (called ParamIC3 in what follows) with the approach described in [8], in order to evaluate the viability of our technique when compared to other SMT-based solutions. For this, we have implemented the algorithm described in [8] using our "regular" SMT-based IC3 implementation as the backend engine used for reachability checking. In what follows, we call this implementation Iterative-Block-Path(IC3). We remark that the tool of [8] was based only on Bounded Model Checking (BMC), and exploited domain-specific information for computing the maximum needed bound, which is not available in our more general context.

In the second part, we evaluate the effectiveness of the optimizations described in the previous section, by comparing the default heuristic used by ParamIC3, using both the full counterexample path π and its initial state $(s_0, 0)$ for blocking bad regions of parameters, with the basic strategy using only $(s_0, 0)$ (called ParamIC3-basic in the following). In particular, the default heuristic used by ParamIC3 works as follows. At the beginning, only initial states $(s_0, 0)$ of counterexample paths are used to block bad regions of parameters. If the algorithm starts enumerating too many bad regions, it starts exploiting also full paths π, by computing the bad region $\beta_k^\pi(U) = \exists X.BMC_k^\pi$, where k is the length of π, and BMC_k^π is the formula encoding all the counterexample traces of length k where the values for the Boolean variables are the same as in π, similarly to what is done in [8]. The computation of β_k^π is aborted if it becomes too expensive,[3] in order to control the tradeoff between the quality of the obtained bad region and the cost of performing quantifier elimination.

Finally, in the third part of our evaluation, we compare ParamIC3 against Red [13], a state-of-the-art tool for parameter synthesis for linear-hybrid automata.

Benchmarks. We have selected benchmarks used in previous work on parameter synthesis for hybrid systems. Most of them come from the suite of Red. We have a total of 92 instances from 13 different families. All the instances, the scripts and the tools used for reproducing our experiments are available at http://es.fbk.eu/people/mover/fmcad13.tar.gz. For the first two parts of our evaluation, we have experimented with two different ways of encoding linear hybrid automata into symbolic transition systems, resulting in a set of 192 instances. For the comparison with Red, we picked the encoding giving the best overall performance for ParamIC3.

Results. We have run our experiments on a cluster of Linux machines with a 2.27GHz Xeon CPU, using a timeout of 600 seconds and a memory limit of 3Gb for each instance. The results are shown in Figures 2–4. From the plots, we can make the following observations. (i) Our new algorithm is clearly superior to the technique of [8], both in number of

[3]We currently use a cutoff value on the number of elementary operations in the quantifier elimination module of MathSAT for this.

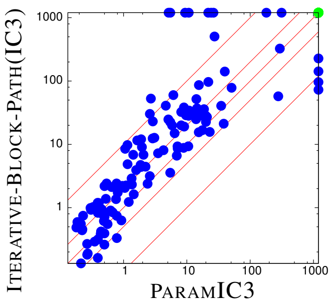

Fig. 2. Comparison between ParamIC3 and Iterative-Block-Path(IC3).

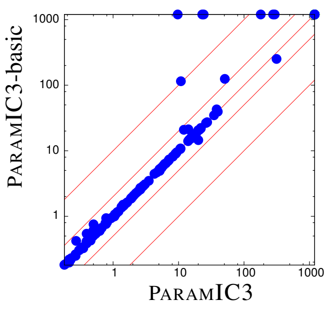

Fig. 3. Comparison between ParamIC3 and ParamIC3-basic.

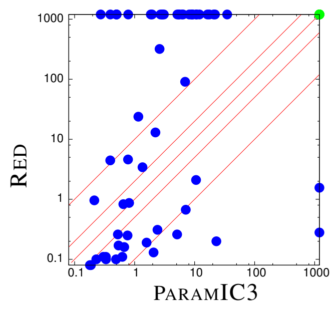

Fig. 4. Comparison between ParamIC3 and Red.

completed instances and in execution time. Overall, ParamIC3 successfully solves 5 more instances than Iterative-Block-Path(IC3), and it is almost always faster. We remark that both algorithms use the same implementation of IC3 as backend, run with the same options. (ii) Our heuristic for using full counterexample paths π for blocking bad regions of parameters pays off for harder problems. With it, ParamIC3 solves 6 more instances which were previously out of reach, without any overhead for the other instances. (iii) The comparison with Red shows that our technique is very promising. Although there is no clear winner, there are more instances for which ParamIC3 outperforms Red than the converse. In general, the two tools seem to be somewhat complementary. We remark that Red is specialized for timed and linear-hybrid automata and that most of the benchmarks we used come from its suite, whereas ParamIC3 works for arbitrary transition systems and it is not tuned for linear hybrid systems in any way.

VI. Conclusions and Future Work

We proposed a new algorithm based on IC3 for synthesizing an optimal region of parameter valuations guaranteeing the satisfaction of an invariant property. The algorithm exploits the features of IC3 to incrementally remove sets of bad parameter valuations and to reduce the cost of expensive quantifier elimination operations by performing them on small formulas. Our experimental results show that the new synthesis algorithm performs better than similar SMT-based techniques and is complementary to other techniques based on the computation of the reachable states. In the future, we plan to improve the algorithm by better exploiting the structure of the problem, to evaluate it in other domains such as software, and to apply it in the context of modular component-based verification.

References

[1] Étienne André, Laurent Fribourg, Ulrich Kühne, and Romain Soulat. IMITATOR 2.5: A tool for analyzing robustness in scheduling problems. In *FM*, pages 33–36, 2012.

[2] Étienne André and Ulrich Kühne. Parametric analysis of hybrid systems using HyMITATOR. In *iFM*, pages 16–19, 2012.

[3] Aurore Annichini, Ahmed Bouajjani, and Mihaela Sighireanu. TReX: A tool for reachability analysis of complex systems. In *CAV*, pages 368–372, 2001.

[4] Aaron R. Bradley. Sat-based model checking without unrolling. In *VMCAI*, pages 70–87, 2011.

[5] Roberto Bruttomesso, Alessandro Carioni, Silvio Ghilardi, and Silvio Ranise. Automated analysis of parametric timing-based mutual exclusion algorithms. In *NASA Formal Methods*, pages 279–294, 2012.

[6] Alessandro Cimatti and Alberto Griggio. Software Model Checking via IC3. In *CAV*, pages 277–293, 2012.

[7] Alessandro Cimatti, Alberto Griggio, Bastiaan Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT Solver. In Nir Piterman and Scott Smolka, editors, *TACAS*, volume 7795 of *LNCS*. Springer, 2013.

[8] Alessandro Cimatti, Luigi Palopoli, and Yusi Ramadian. Symbolic computation of schedulability regions using parametric timed automata. In *RTSS*. IEEE Computer Society, 2008.

[9] Goran Frehse, Sumit Kumar Jha, and Bruce H. Krogh. A counterexample-guided approach to parameter synthesis for linear hybrid automata. In *HSCC*, pages 187–200, 2008.

[10] Thomas A. Henzinger and Pei-Hsin Ho. Hytech: The cornell hybrid technology tool. In *Hybrid Systems*, pages 265–293, 1994.

[11] Krystof Hoder and Nikolaj Bjørner. Generalized property directed reachability. In *SAT*, pages 157–171, 2012.

[12] Roland Kindermann, Tommi A. Junttila, and Ilkka Niemelä. Smt-based induction methods for timed systems. In *FORMATS*, pages 171–187, 2012.

[13] Farn Wang. Symbolic parametric safety analysis of linear hybrid systems with bdd-like data-structures. *IEEE Trans. Software Eng.*, 31(1):38–51, 2005.

APPENDIX

We prove Theorem 1, which states that the region of parameters ρ found by ParamIC3 is optimal (and thus, correct). The proof is by induction on the number of iterations of ParamIC3. The proof shows that if the algorithm terminates, it returns the optimal region of parameters.

Consider the first iteration of the algorithm.

Suppose we prove $S_\rho \models P$. Then, we know that there are no bad parameter valuations such that S_ρ is not safe. Also, $\rho = \top$ is the optimal parameter region.

Instead, suppose we find $\pi = s_0; \ldots ; s_n$, a counterexample to $S_T \models P$. The counterexample π found by IC3 is such that for each state u in s_0 there exists a path $u_0, \ldots , u_n$, where $u_n \models \neg P$. We compute the set of bad parameters $\beta_{s_0}(U) := \exists X.s_0(U, X)$. From π, we know that for all parameter valuations $\gamma \in \beta_{s_o}$, there exist a state u_0 in s_0 (i.e. $u_0 \models s_0$) and a path $u_0; \ldots ; u_n$ such that $\gamma \models u_0$ and $u_n \models \neg P$. Thus, $S_{\rho_\gamma} \not\models P$ for all $\gamma \in \beta s_0$. Intuitively, it means that ParamIC3 only removes valuations of unfeasible parameters.

The new under-approximation of bad parameters is $\beta' := \beta(s_0)$, while the over-approximation of the good parameters is $\rho' := \rho \wedge \neg\beta(s_o)$. We consider the transition system $S_{\rho'}$ and the new problem $S_{\rho'} \models P$. Then, the algorithm set the following variables (here, we use the primed notation to refer to the value of the variables after the assignment): $I' := I \wedge \neg\beta(s_o)$, $T' := T \wedge \neg\beta(s_0)$, $F_0' := I \wedge \neg\beta(s_o)$. The invariants on the IC3 traces (the one presented in the Subection II-C) holds for checking the problem $S_{\rho'} \models P$:

- $F_0' = I_{\rho'}$: it holds, since $F_0' := I \wedge \neg\beta(s_o) = I_{\rho'}$;
- $F_i' \models F_{i+1}$: it holds, since $F_0' \models F_1$;
- $F_0' \models F_1$ and for $1 \le i \le n$, $F_i(X) \wedge T_{\rho'}(X, X') \models F_{i+1}(X')$.
  Consider $F_0'(X) \wedge T_{\rho'}(X, X') \models F_{i+1}(X')$.
  We have to prove that $F_0'(X) \wedge T_{\rho'}(X, X') \wedge \neg F_{i+1}(X') \models \bot$. By hypothesis, we know that $F_0(X) \wedge T_\rho(X, X') \wedge \neg F_{i+1}(X') \models \bot$. Suppose there exists a model μ such that $\mu \models F_0'(X) \wedge T_{\rho'}(X, X') \wedge \neg F_{i+1}(X')$. Recall that $F_0'(X) \wedge T_{\rho'}(X, X') \wedge \neg F_{i+1}(X')$ is $F_0(X) \wedge (\neg\beta_{s_0}) \wedge T_\rho(X, X') \wedge (\neg\beta_{s_0}) \wedge \neg F_{i+1}(X')$. Thus, $\mu \models F_0(X) \wedge T_\rho(X, X') \wedge \neg F_{i+1}(X')$, contraddicting the hypothesis.
  By a similar reasoning, we can prove that $F_i(X) \wedge T_{\rho'}(X, X') \models F_{i+1}(X')$ holds, for $1 \le i \le n$.
- for all $i < k$, $F_i \models P$: it holds, since $F_0' \models P$.

Now, suppose we are at the $n-th$ iteration, where ρ_n and β_n are the approximations of good and bad parameters found so far.

- Suppose we prove $S_{\rho_n} \models P$. Thus, we proved that ρ_n is a region of good parameters.
  Note that, by induction every bad region found by ParamIC3 $\beta_1 \wedge \ldots \beta_n$ contains only unfeasible parameter valuations and $\rho_n := \neg(\beta_1 \vee \ldots \vee \beta_n)$. Thus, ρ_n is optimal.
- Instead, suppose we still find a counterexample $\pi = s_0; \ldots ; s_n$. We compute $\beta(s_0)$, which is a set of bad parameter valuations for S_{ρ_n}. We can prove that $\beta(s_0)$ only contains valuations for "bad" parameters applying the same reasoning done in the first iteration. Note that, since S_{ρ_n} under-approximate $S_\top$, $\beta(s_0)$ is bad region also for the parametric transition system.
  We have that $S_{\rho_{n+1}} = S_{\rho_n} \wedge (\neg\beta(s_0))$. With a resoning similar to the one that we did in the fist iteration case, we can prove that the invariant on the frames of IC3 holds for the model checking problem $S_{\rho_{n+1}} \models P$.