Challenges in Constraint-based Analysis of Hybrid Systems*

Andreas Eggers[1], Natalia Kalinnik[2], Stefan Kupferschmid[2], and Tino Teige[1]

[1] Carl von Ossietzky Universität Oldenburg, Germany
{eggers|teige}@informatik.uni-oldenburg.de
[2] Albert-Ludwigs-Universität Freiburg, Germany
{kalinnik|skupfers}@informatik.uni-freiburg.de

Abstract. In the analysis of hybrid discrete-continuous systems, rich arithmetic constraint formulae with complex Boolean structure arise naturally. The iSAT algorithm, a solver for such formulae, is aimed at bounded model checking of hybrid systems. In this paper, we identify challenges emerging from planned and ongoing work to enhance the iSAT algorithm. First, we propose an extension of iSAT to directly handle ordinary differential equations as constraints. Second, we outline the recently introduced generalization of the iSAT algorithm to deal with probabilistic hybrid systems and some open research issues in that context. Third, we present ideas on how to move from bounded to unbounded model checking by using the concept of interpolation. Finally, we discuss the adaption of some parallelization techniques to the iSAT case, which will hopefully lead to performance gains in the future. By presenting these open research questions, this paper aims at fostering discussions on these extensions of constraint solving.

Keywords: mixed Boolean and arithmetic constraints, differential equations, stochastic SMT, Craig interpolation, parallel solver.

1 Introduction

The complexity of embedded systems, e.g. in automotive and avionics applications, has increased dramatically over the last decades. The safety criticality of these systems calls for more and more sophisticated — especially computer-aided — analysis techniques that enable engineers to assess the correctness of their designs and implementations. For finding errors in models of large systems, simulation has become one of the most successful and established methods. However, in general, simulation cannot guarantee the absence of errors for systems with infinitely many states which naturally arise in these domains.

In recent years, algorithms have been developed that can mathematically prove the correctness of a huge variety of system classes with respect to a given

* This work has been partially supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, www.avacs.org).

specification. Embedded systems often combine digital and analog components, e.g. in multi-modal controllers or when describing them as integrated models of a digital controller interacting with its continuously evolving plant. *Hybrid systems* are a very rich modelling paradigm to describe such hybrid discrete-continuous behavior. A hybrid system consists of a set of modes and a set of continuous variables that together represent its state space. Its evolution is described by a transition relation entailing discrete mode switches, also called transitions, and arithmetic constraints describing the behavior of the continuous variables within each mode. The latter is often achieved by using differential equations that naturally arise when modelling physical entities. The mode switches are governed by so-called transition guards, i.e. arithmetic constraints observing the continuous variables, and can perform discrete actions by (potentially non-deterministically) setting a variable x to a new value x' satisfying an arithmetic condition, e.g. $x' > \sin(y^2)$ or $x' = 4.2 \cdot x$.

The semantics of a hybrid system is defined by the set of its runs, i.e. the possible evolutions it allows. Such an evolution can always be represented by a sequence of variable valuations, where two successive valuations can either be connected by a continuous evolution in the mode the system is in, or satisfy the transition guard and action constraints, such that the system can actually perform a switch from one mode to the next. This representation of a run is called a trace and intuitively describes snapshots of the system's evolution through the state space at the endpoints of continuous trajectories and discrete jumps.

Returning to the motivation described initially, the reachability problem of hybrid systems, i.e. the question of whether a particular state (e.g. a state representing a fatal system failure) is reachable, is of particular interest to complex systems verification and falsification. Though this problem is undecidable in general, developing model checking algorithms and tools that can deal with a large sample of systems that occur in real-world applications seems to be so relevant that it can be considered a reasonable goal nonetheless. In addition to that, robustness notions [1] can be used to find classes of systems, for which decision procedures can be developed. Hybrid systems and decidability questions have been extensively examined in the literature, for a detailed account see e.g. [2].

Among the most successful analysis methods for finite-state systems is *bounded model checking* (BMC) [3, 4], which has also been extended to the case of hybrid systems [5, 6]. The idea of BMC is to encode the initial states, the transition relation, and the target state specification of the system as predicates $\mathit{INIT}(\boldsymbol{x}_0)$, $\mathit{TRANS}(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$, and $\mathit{TARGET}(\boldsymbol{x}_k)$, respectively, where $\boldsymbol{x}_0, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}$, and $\boldsymbol{x}_k$ are instantiations of the vector of variables representing the discrete and continuous state space. The initial predicate $\mathit{INIT}(\boldsymbol{x}_0)$ is satisfied by a valuation of $\boldsymbol{x}_0$ iff that valuation characterizes an initial state. Analogously, the transition predicate $\mathit{TRANS}(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$ holds for two (successive) valuations iff the system can perform a discrete mode switch or a continuous evolution as described above. We consider the succession from $\boldsymbol{x}_i$ to $\boldsymbol{x}_{i+1}$ as a *step* of the system. Finally, the target predicate $\mathit{TARGET}(\boldsymbol{x}_k)$ specifies the states whose reachability is examined. A hybrid system can thus reach a target state in a limited number of

steps k iff the following BMC formula is satisfiable.

$$\Phi_k := \mathit{INIT}(\boldsymbol{x}_0) \wedge \underbrace{\mathit{TRANS}(\boldsymbol{x}_0, \boldsymbol{x}_1) \wedge \ldots \wedge \mathit{TRANS}(\boldsymbol{x}_{k-1}, \boldsymbol{x}_k)}_{k \text{ unwindings of the transition relation}} \wedge \mathit{TARGET}(\boldsymbol{x}_k)$$

As the behavior of a hybrid system can in general be arbitrarily non-linear and non-deterministic, the resulting BMC formula Φ_k is a Boolean combination of rich arithmetic constraints including differential equations. A solver that can directly handle Φ_k is thus desirable. Approaches from continuous constraint programming (cf. e.g. [7]) which can handle non-linear constraints are often restricted to conjunctive formulae. On the other hand, most *satisfiability modulo theories* (SMT, e.g. [8]) solvers — though being very capable of handling complex Boolean structure — are confined to decidable theories — in particular, they do not handle non-linear constraints. Recently, algorithms combining both domains were published: ABsolver [9], which uses a non-linear optimization packet, and iSAT [10], which uses techniques from interval constraint solving.

Structure of the paper. In Section 2, we briefly recall the iSAT algorithm that constitutes the basic framework for our presentation. Section 3 identifies the main directions for the extensions discussed in this paper, which are described in more detail in Sections 4 – 7. Finally, we give some thoughts on synergies between these different topics.

2 The iSAT algorithm

In [10], the iSAT algorithm for solving mixed Boolean and non-linear (including transcendental) arithmetic constraints over bounded reals and integers was introduced. Differential equations however cannot be handled directly by iSAT and need to be (over-)approximated or solved during modeling. Internally, iSAT solves a conjunction of clauses, where a clause is a disjunction of atoms. An atom (a.k.a. primitive constraint) contains one relational operator, at most one arithmetic operation, and up to three variables, e.g. $x \geq \sin(y)$, $x = y+z$, and $z < 3.7$. By a Tseitin-like transformation [11], any BMC formula Φ_k can automatically be rewritten into an equi-satisfiable formula in this kind of conjunctive normal form, which grows at most linearly in the size of the original formula. The iSAT algorithm is a generalization of the Davis-Putnam-Logemann-Loveland (DPLL) procedure [12, 13] using interval constraint propagation (ICP) (cf. e.g. [7]), and manipulates interval valuations of the variables by alternating *deduction* and *splitting* phases.

During the *deduction* phase, the solver searches for clauses in which all but one atom are inconsistent under the current interval valuation. These remaining consistent atoms are called *unit*. In order to retain a chance for satisfiability of the formula, unit atoms have to be satisfied. This is similar to Boolean constraint propagation in DPLL SAT solving. The unit atoms are therefore used for ICP during the deduction phase. New interval bounds can thus be deduced until a

fixed point is reached. For termination reasons, the ICP has to be stopped if the progress of newly deduced bounds becomes negligible.

If a *conflict* occurs, i.e. the interval of a variable becomes empty, then a conflict resolution procedure is called which analyzes the reason for the conflict. If the conflict cannot be resolved the given formula is unsatisfiable. Otherwise, a conflict clause is built from the reason of the conflict and added to the formula in order to prevent the solver from revisiting the same conflict again. Furthermore, conflict resolution requires the solver to take back some of the decisions and their accompanying deductions that have been performed so far.

If the solver finds a *solution*, i.e. at least one atom in each clause is satisfied by every point in the interval valuation, the algorithm stops. In general, equations like $x = y \cdot z$ can only be satisfied by point intervals. However, reaching such point intervals by ICP cannot be guaranteed for continuous domains. One option to mitigate this problem is to stop the search when all intervals have a width smaller than a certain threshold, the so-called *minimum splitting width*, and returning the found *approximative* solution. Having completed the deduction phase and neither found a conflict nor an (approximative) solution, iSAT performs a decision by *splitting* an interval. A decision heuristics is used to select one of the intervals whose width is still greater than or equal to the minimum splitting width. The search is then resumed using this newly generated interval bound which potentially triggers new deductions as described above.

3 Problem description

The primary goal of this paper is to propose challenges that arise from planned and ongoing work in the context of enhancing the iSAT algorithm into the directions of scope and performance. We hope that presenting these research questions will foster discussions on these interesting topics.

In order to avoid an a priori overapproximation of the continuous dynamics in system models, direct handling of ordinary differential equations is to be integrated into iSAT's deduction rules (Section 4). Another extension of the scope is to enable iSAT to deal with probabilistic hybrid systems (Section 5). Thereafter, we present ideas on how to move from bounded to unbounded model checking by using the concept of interpolation. In Section 7, we discuss the adaption of some parallelization techniques to the iSAT case, which will hopefully lead to dramatic performance gains in the future.

4 Differential equations as constraints

In order to directly handle ordinary differential equations (ODEs) as constraints within a formula, we need to extend the deduction mechanism used by iSAT to not only propagate newly deduced bounds through arithmetic constraints using ICP, but to also propagate new interval bounds through ODEs. A continuous trajectory can often be described by an ODE of the form $\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x})$ over a vector $\boldsymbol{x}$ of continuous variables. Being part of the predicative encoding of the

transition relation, this ODE describes the connection of the variable instances $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$ from two successive BMC unwinding depths. We thus search for solution functions of the ODE that emerge from the valuation of $\boldsymbol{x}_i$ (the *prebox*) and eventually reach the valuation of $\boldsymbol{x}_{i+1}$ (the *postbox*). Analogously to ICP, we are interested in pruning off all valuations from the pre- and postbox that cannot belong to such trajectories. To achieve this, we thus need a safe overapproximation of the ODE trajectories in order not to prune away possible solutions.

Related work on safe enclosures of ODEs can be found in [14–16], where error bounds on the remainder term of a *Taylor series* of the unknown exact solution are calculated and used as safe overapproximations of the errors induced by the Taylor-series-based approximation of the trajectory. Using coordinate transformations to suitably adapt the enclosures to the solution sets and thereby mostly avoiding the so-called *wrapping effect* (cf. e.g. [14]), this approach works well for linear ODEs. However, as for non-linear ODEs, coordinate transformations alone are often insufficient to eliminate the wrapping effect, the enclosures quickly become very rough and finally unusably large in the non-linear case. A newer approach — the so-called *Taylor models* [17] — have been shown to give tighter enclosures for non-linear ODEs by employing a more symbolic representation of the enclosure sets. Henzinger et. al. use the Taylor-series-based enclosure method in the HyperTech tool [18], also facing hybrid systems analysis, however not in a constraint solving approach. In [19], CLP(F) — a very broad framework to constraint logic programming — is applied to models of hybrid systems. CLP(F) does however not include any measures against the wrapping effect encountered when enclosing ODE trajectories with intervals.

This section of the paper tries to sum up the essential challenges and options to solve them, that we see in the context of embedding safe enclosures of ODEs into the iSAT algorithm. These major challenges are to

1. find methods and data structures that allow sufficiently tight overapproximating enclosures of the trajectories of an ODE that connect the interval boxes representing pre- and postsets,
2. devise heuristics that allow to select the method fitting best into the current phase of solving, e.g. coarse-grain but quick first enclosures to chop off the most implausible parts of the solution space during an early phase of solving versus tight but expensive enclosures to narrow an enclosure tube around an actual error trace to reduce the probability of spurious counterexamples,
3. embed these methods in the solver process anywhere between calling them like normal deduction rules that are executed whenever a new bound on a variable is generated or as a subordinate solver that can be called arbitrarily seldom to reduce the impact of an expensive method,
4. derive symbolic knowledge from ODE constraints that can be learned and thus automatically extend the constraint system to contain redundant encodings reflecting some of the possible system dynamics without the need to (probably always more expensively) re-enclose the ODE trajectories.

In a first prototype, we have proved the feasibility of integrating a Taylor-series based enclosure method as a subordinate solver to the iSAT algorithm. However, first experiments with this prototype show that none of above challenges can be regarded as completely mastered [20]. For each challenge a multitude of design choices exist that may have a fundamental impact on the overall performance of an ODE-enabled iSAT.

To approach the first challenge, we consider both the Taylor-based enclosure methods [14–16], that were used for the prototype, and the more recently developed Taylor models [17] as possible choices. While Taylor series together with coordinate transformations will probably be a good choice for linear ODEs, we expect Taylor models to also allow to approach non-linear ODEs. Out of the many existing numerical methods for numerical approximation of ODE trajectories there may however be some whose truncation errors can be enclosed as well. Exploring such methods may thus extend the spectrum of choices. The most essential problem in this context will probably be to control the wrapping effect in order to avoid unusably coarse overapproximations.

The second challenge necessitates, first, a pool of methods with different characteristics, i.e. methods that are tailored to quickly generating results as well as methods that allow tight enclosures, and second, a set of criteria that are easily evaluable and allow to determine which enclosure method should be used. One possible criterion could be the size of the currently searched box, where a small box could indicate the use of more accurate methods.

Solving the third challenge will mean to find the best integration strategy for the enclosure mechanisms. It can be expected that any good enclosure method will normally be quite expensive in terms of runtime compared to arithmetic interval propagators. This may suggest that performing enclosures very seldom might be a good strategy. On the other hand, as an essential portion of the system dynamics is encoded in the ODE constraints, it also seems necessary to evaluate them often in order to detect conflicts early and thus to prune off those parts of the search space that cannot contain any solutions.

Finally, we expect that learning new arithmetic constraints from the ODEs (e.g. from monotony or stability arguments) will allow to reduce the number of enclosures that actually need to be performed. Similar to learning conflict clauses when the intersection of an enclosure with a pre- or postbox becomes empty, these constraints would allow to prune off substantial parts of the search space that cannot contain any solutions.

5 Stochastic constraint systems

Most of the common analysis procedures are confined to just prove or disprove the safety of a system. However, for models of safety-critical systems interacting with the environment it is often clear which incidents lead to unsafe behavior, e.g. a power blackout combined with a failure of the emergency power system can induce a safety-critical state of a nuclear power station. Although such accidents cannot be excluded in general, it is tried to strongly decrease the prob-

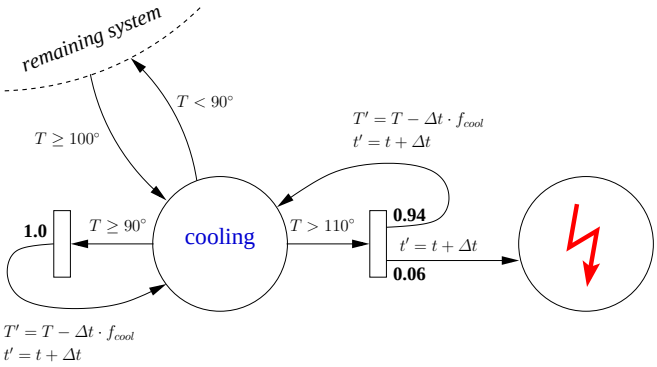

Fig. 1. A fragment of a system model for a probabilistic component breakdown

ability of safety-critical behavior s.t. the system is *most likely* safe. Thus, the verification goal in this application domain is whether the *likelihood* of reaching unsafe states is below an acceptable threshold, e.g. less than 0.03%. As a modeling framework to deal with uncertainties, we consider *probabilistic hybrid automata* (PHA, cf. [21]) which extends the notion of hybrid automata s.t. the non-deterministic selection of a transition is enriched by a probabilistic choice according to a distribution over variants of the transition. I.e., each transition carries a (discrete) probabilistic distribution. Each probabilistic choice within such a distribution leads to a potentially different successor mode while performing some discrete actions. An example of a PHA fragment modelling some probabilistic component breakdown is shown in Fig. 1, where T, f_{cool}, and t denote the temperature, the cooling factor, and the time, resp., and Δt is the discretization parameter.

In order to automatically compute the reachability probability of (un-)desired properties of PHAs, in [21] we introduced *stochastic satisfiability modulo theories* (SSMT) as the unification of stochastic propositional satisfiability (SSAT) [22] and satisfiability modulo theories (SMT, e.g. [8]). SSMT deals with *existential* and *randomized* quantification of finite-domain variables. An SSMT formula is specified by a quantifier prefix and an SMT formula, e.g. $\exists x \in \{0,1\}\ \mathfrak{U}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} : (x > 0 \vee \sin(4a) \geq 0.3) \wedge (y > 0 \vee \sin(4a) < 0)$. The value of a variable bound by an existential quantifier, as in $\exists x \in \{0,1\}$, can be set arbitrarily, while the value of a variable bound by a randomized quantifier, as in $\mathfrak{U}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}$, is determined stochastically by the corresponding distribution, here $\langle(0,0.6),(1,0.4)\rangle$. E.g., $\mathfrak{U}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}$ means that the variable y is assigned the value 0 or 1 with probability 0.6 or 0.4, resp. The solution of an SSMT problem Φ is a tree of assignments to the existential variables that *maximizes the overall satisfaction probability* of Φ (cf. [21] for more details). In our application, we are interested in computing the maximum probability of satisfaction. For the example above, setting x to 0 yields the satisfaction probability 0.4 since the assignment $x = 0, y = 0$ cannot satisfy the SMT for-

mula. For $x = 1$, both $y = 0$ and $y = 1$ lead to solutions and, thus, to satisfaction probability 1.0. Hence, the maximum satisfaction probability is 1.0.

The behavior of a PHA $\mathcal{H}$ (restricted to step depth k) together with a reachability property P can be described by an SSMT formula Φ in the following sense: Φ is satisfiable with maximum probability p iff $\mathcal{H}$ fulfills property P (within k steps) with maximum probability p. The idea of the formalized encoding of a PHA into an SSMT formula, as presented in [21], is that the non-deterministic choice of a transition in a PHA corresponds to existential quantification in SSMT, while the probabilistic distributions correspond to randomized quantification. The discrete-continuous behavior of the automaton is encoded by means of standard techniques. We are currently working on a modeling framework for PHAs which automatically translates the modelled PHA into an SSMT formula [23].

Completing the verification procedure for PHAs, we recently extended the iSAT solver to existential and randomized quantification (SiSAT, [24]). The main idea of the SiSAT algorithm is to traverse the tree given by the quantifier prefix and to properly call the iSAT algorithm. First experimental results proved the concept: exploiting additional pruning rules which cut off huge parts of the quantifier tree, the SiSAT tool is currently able to solve SSMT problems with up to 110 quantified and 350 non-quantified variables, and up to 1100 clauses within 100 minutes. Problems including quantifiers are well-known not to be as scalable as quantifier-free problems. However, we believe that further improvements, e.g. value and variable ordering heuristics, will yield significant performance gains. In the following, we highlight some open research issues for future work.

- *Value and variable ordering heuristics* are well-studied in SAT and Constraint Satisfaction to improve efficiency. For the quantified case, the variable ordering within a block of quantifiers with the *same* type do not change the semantics of the problem. This property can be exploited during the proof search to rearrange the variables. To derive benefit from this fact, we will investigate different static and dynamic ordering heuristics.
- *Bounded model checking*, i.e. stepwise unrolling the transition relation of a system interspersed with model checking runs, facilitates to reuse and propagate knowledge from previous runs (due to symmetries) such as *conflict clause reusing and shifting*. In the quantified case, we are also interested in maintaining and propagating knowledge about *solutions* of previous solver calls. Besides skipping branches leading to a conflict, such a technique would allow to avoid investigation of branches for which the satisfaction probability was (partially) computed previously.
- The underlying iSAT algorithm which is based on interval arithmetic is in general neither able to find a (real) solution nor to prove its absence. In such cases the results are approximative solutions which suffice certain consistency notions but do not guarantee real solutions. Concerning the reliability of the computed satisfaction probabilities, we will extend the SiSAT tool to deal with confidence intervals in order to obtain *safe approximations of satisfaction probabilities.*

- Another issue concerning the reliability of the computed results is to offer *certificates of the satisfaction probabilities*, i.e. proofs that the returned probabilities are correct. Reliable results are of utmost importance in the verification of safety-critical industrial systems. A certificate that a quantifier-free formula is satisfiable is simply a satisfying assignment to the variables. A proof of unsatisfiability is often more complex, e.g. a clause resolution strategy in the propositional case. In our setting, such a certificate seems to be a mixed version of both.
- To assess the practical significance, we will apply the SiSAT tool on *real-world benchmarks*. Within the AVACS project[3], we are especially interested in benchmarks which deal with the impact of cooperative, distributed traffic management on flow of road traffic. These benchmarks are representative for a large number of hard scheduling and allocation problems and naturally show uncertain behavior.
- A more fundamental challenge is to generalize the scope of the quantifiers to continuous domains involving arbitrary probability distributions. This would increase considerably the expressive power of SSMT.

6 Interpolation

In system's verification, i.e. unbounded model checking, Craig interpolants [25] have gained more and more attention over the last years. In [26], McMillan modified a bounded model checking procedure for Kripke Structures with the help of interpolants s.t. the procedure becomes able to prove safety properties of a given system for runs of *any* length. More recently, McMillan extended his work on unbounded model checking to the quantifier-free theory of linear inequality and uninterpreted function symbols [27], which is used, e.g., in software verification. His approach has been implemented in the software model checker Blast [28].

As outlined in Section 1, bounded model checking aims at *disproving* a property $P(\boldsymbol{x})$ of a given system $\mathcal{S}$. Thus, a BMC procedure tries to find an unwinding depth k s.t. the corresponding BMC formula Φ_k with $TARGET(\boldsymbol{x}_k) = \neg P(\boldsymbol{x}_k)$ is satisfiable. On the other hand, the goal of *unbounded* model checking is to prove that P holds for all runs of $\mathcal{S}$. I.e., Φ_k with target $\neg P(\boldsymbol{x}_k)$ is unsatisfiable for any $k \in \mathbb{N}$.

Such an unbounded model checking procedure can be obtained by means of *Craig interpolation*. Given two formulae A and B s.t. $A \wedge B$ is unsatisfiable. Then, a formula p is called *Craig interpolant for A and B* iff (1) p contains only variables which occur in both A and B (*(A,B)-common variables*), (2) $A \Rightarrow p$, and (3) $p \Rightarrow \neg B$. A Craig interpolant p is called *strongest* if p implies any other Craig interpolant p', i.e. $p \Rightarrow p'$. Hence, any such interpolant p' overapproximates p.

After showing that $\Phi_0 = INIT(\boldsymbol{x}_0) \wedge TARGET(\boldsymbol{x}_0)$ is unsatisfiable (i.e. initially the target does not hold), McMillan's procedure first solves the BMC

[3] www.avacs.org

formula $\Phi_1 = PREF \wedge SUFF$, where

$$PREF := REACH(\boldsymbol{x}_0) \wedge TRANS(\boldsymbol{x}_0, \boldsymbol{x}_1) \text{ and}$$
$$SUFF := TARGET(\boldsymbol{x}_1),$$

and initially $REACH(\boldsymbol{x}_0) := INIT(\boldsymbol{x}_0)$. If Φ_1 is unsatisfiable then a Craig interpolant $p(\boldsymbol{x_1})$ for the formulae $PREF$ and $SUFF$ is computed.[4] By $PREF \Rightarrow p(\boldsymbol{x_1})$, the interpolant $p(\boldsymbol{x_1})$ is an overapproximation of the states reachable in one system step from $REACH(\boldsymbol{x}_0)$. If this overapproximation shifted to the zeroth instantiation of the variables (as described by $p(\boldsymbol{x_0})$) is a subset of the so far reachable states, i.e. $p(\boldsymbol{x_0}) \Rightarrow REACH(\boldsymbol{x}_0)$, then further transitions can only lead to states already characterized by $REACH(\boldsymbol{x})$. As a consequence, the target states are unreachable and the verification procedure succeeds. Otherwise, we expand the set of reachable states s.t. it also covers the reachable states given by the shifted interpolant, i.e. $REACH(\boldsymbol{x}_0) := REACH(\boldsymbol{x}_0) \vee p(\boldsymbol{x_0})$. Then, the procedure is iterated until the above termination criterion holds. Due to the overapproximations of the reachable state set, showing the satisfiability of one of the obtained formulae Φ_1 does not imply that the target state is actually reachable. For a more detailed account, confer [26].

Computing Craig interpolants for different theories can be found in the literature [27, 29, 30]. However, none of these approaches is capable of constructing interpolants for the case of mixed Boolean and non-linear arithmetic constraints including transcendental functions. Therefore, extending the concept of interpolation to this richer domain originating from hybrid systems analysis is an interesting research issue. In the sequel, we identify the most essential challenges:

- Obtaining interpolants in the iSAT case requires construction rules. One way might be to generalize Pudlák's algorithm [31], that delivers interpolants for the propositional case using the proof of unsatisfiability.
- Craig interpolants are not unique and therefore there exist interpolants that are bigger or smaller, stronger or weaker then others, etc. Thus, an open problem is to determine which characteristics of interpolants are favorable especially in the sense of low computational costs.
- As the reachability problem of hybrid systems is in general undecidable, it is worthwhile to identify decidable subclasses for which the interpolation procedure always terminates. One promising starting point is to investigate *robustness* notions for hybrid systems (cf. e.g. [1]), which may guarantee such a termination property.

The following example illustrates that selecting suitable Craig interpolants is a difficult problem. The system $\mathcal{S}$ with $INIT(\boldsymbol{x}_0) = x_0 \geq 1$, $TRANS(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) = x_{i+1} \geq 0.5x_i$, and $TARGET(\boldsymbol{x}_k) = x_k \leq 0$ describes the evolution of a variable x that is initially greater than 1 and is iteratively divided by 2. A property of $\mathcal{S}$ is that x will never become less than 0. Consider the formula $\Phi_1 = PREF \wedge SUFF = (x_0 >= 1 \wedge x_1 = 0.5x_0) \wedge (x_1 \leq 0)$ which is unsatisfiable. A possible

[4] Note that $p(\boldsymbol{x_1})$ may only contain $(PREF, SUFF)$-common variables.

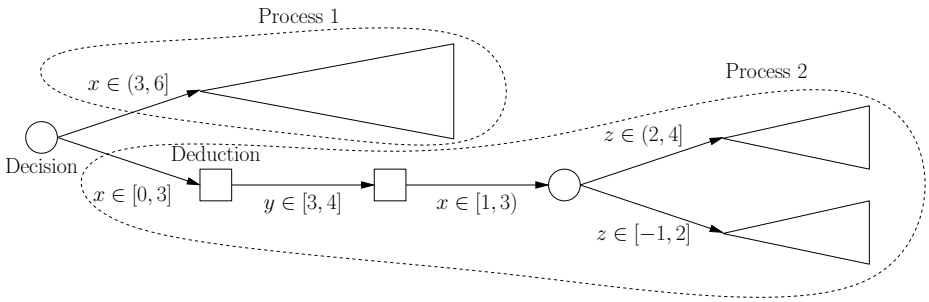

Fig. 2. Search space partitioning at interval splitting points in iSAT (two processors)

Craig interpolant is $p^1 = x_1 \geq 0.5$. As $p^1 \not\Rightarrow \mathit{INIT}(\boldsymbol{x}_0)$ we use p^1 as the new initial state. The resulting formula $(x_0 >= 0.5 \wedge x_1 = 0.5x_0) \wedge (x_1 \leq 0)$ is also unsatisfiable. A possible Craig interpolant is $p^2 = x_1 \geq 0.25$. Since p^2 does not imply p^1, we proceed. If we had computed $p^1 = x_1 > 0$ then p^2 would imply p^1, resulting in a fixed point. Though the example suggests that weaker interpolants are more suitable than stronger ones, this needs not to be true in general.

7 Parallelization

Recent trends in hardware design towards multi-core and multiprocessor systems, and computer clusters call for the development of dedicated parallel algorithms in order to exploit the full potential of these architectures. In the domain of propositional SAT solving, parallel algorithms can be traced back to at least 1994, when Böm and Speckenmeyer presented the first parallel implementation of a DPLL procedure for a transputer system consisting of up to 256 processors [32]. During the past decade, more advanced implementations have been developed. Most existing parallel SAT solvers are based on DPLL, they are, however, parallelized in different ways and focus on different hardware environments.

While some of them, such as PaSAT [33], PaMira [34], Satz [35], are designed for distributed memory systems, others, like ySAT [36], MiraXT [37], are tailored to use shared memory workstations. Both shared-memory and distributed-memory workstations have advantages and disadvantages. Shared memory computers have the benefit that all processors can access a shared common address space and guarantee in general low latency and low communication overhead. In distributed systems, on the other hand, each processor has its own local memory. Hence, processors communicate over the network via messages causing slow inter-process communication. Choosing the right memory architecture has thus an important impact on the performance of any parallelized algorithm.

As iSAT builds upon DPLL, adapting different parallelization approaches from purely propositional SAT solving to this richer framework constitutes an important first goal. In [38], *guiding paths* are used to partition the search space of a propositional SAT problem into non-overlapping parts that can be treated

in parallel by dynamically allotting them to different processors. The underlying idea is to split the search space at the decision points of the DPLL search tree, i.e. at points where a value for a propositional variable is selected. For this purpose, the guiding path keeps track of possible alternative decisions that can be given to an idle processor. This concept can be adapted to the iSAT context by partitioning the search space at interval-splitting points (cf. Fig. 2).

Furthermore, the exchange of conflict clauses is an essential ingredient of parallel SAT solvers to gain performance. Each conflict clause describes a part of the search space which does not contain any solution. Thus, exchanging conflict clauses prevents other processes from examining such conflicting parts that have already been proven unsatisfiable by another process. Another such element is to employ different decision heuristics for each involved processor. In [34], it was shown that selecting the variables according to different decision heuristics accelerated the PaMira solver by 70% on average.

In addition to parallelization techniques from propositional SAT, the iSAT algorithm introduces new options. As the deduction mechanisms in iSAT (e.g. ICP) are in general more expensive than Boolean deductions, parallelizing iSAT's deduction phase could be beneficial. Another observation is that smaller values for the *minimum splitting width* (cf. Section 2) typically cause longer runtimes of iSAT but allow more precise results. Exploiting this, solver instances with greater minimum splitting widths could supply those instances having smaller widths with conflict clauses in order to accelerate their search.

The high computational costs of checking large BMC instances call for the development of parallel BMC techniques. While some approaches apply parallel solvers to the same BMC instance, the authors of [39] introduce a different approach by simultaneously solving different BMC instances. Moreover, they also adapt the concept of sharing and shifting conflict clauses, first proposed by Strichman [40] for sequential BMC, to the parallel setting. Since BMC formulae Φ_k and Φ_m for the same system share common subformulae, it makes sense to exchange conflict clauses between the corresponding solver instances. Shifting conflict clauses is a related technique, that exploits the symmetry between different BMC formulae originating from the same system. As a BMC formula Φ_k consists of k instantiations of the transition relation, conflict clauses can be shifted within the current instantiation depth k. It is an open question whether a similar parallel BMC scheme for non-linear hybrid systems with iSAT as the underlying constraint solver yields performance gains comparable to those encountered for linear hybrid systems using a combined SAT-LP solver [39].

8 Conclusion

In this paper, we sketched a number of challenges emerging from ongoing work on the constraint-based analysis of hybrid systems. While these extensions are currently developed separately from each other, core technologies like ICP or conflict analysis are used commonly. We hope that by keeping these developments closely together, in particular by sharing data structures, synergies be-

come accessible in the long run. We think that among the issues emerging from integration, some are more obvious than others. For example, employing ODE deduction mechanisms as a subordinate solver within SiSAT or the parallelized iSAT seems to be unproblematic. The same holds for the usage of a parallelized solver as a decision engine within the stochastic SMT algorithm or the interpolation approach. On the other hand, even the theory of interpolation within a probabilistic environment is still unclear, as is the generation of interpolants from formulae comprising ODE constraints.

While some details presented in this paper are specific to the iSAT context, we think that the broader issues are of more general interest. For instance, ODE propagation could be used within other SMT approaches [8] as a theory solver, while e.g. decision heuristics and certificate generation may not only be applicable to SSMT but could also be used in stochastic constraint programming [41].

Acknowledgements

The authors would like to thank Erika Ábrahám, Bernd Becker, Christian Herde, Holger Hermanns, and Tobias Schubert for many valuable discussions about the presented topics, and Martin Fränzle for additionally commenting on earlier versions of this paper. Furthermore, the authors are very grateful to the anonymous reviewers for their helpful remarks.

References

1. Fränzle, M.: Analysis of hybrid systems: An ounce of realism can save an infinity of states. In Flum, J., Rodriguez-Artalejo, M., eds.: Computer Science Logic (CSL'99). Number 1683 in LNCS, Springer Verlag (1999) 126–140
2. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? In: Proc. of the 27th Annual Symposium on Theory of Computing, ACM Press (1995) 373–382
3. Groote, J.F., Koorn, J.W.C., van Vlijmen, S.F.M.: The Safety Guaranteeing System at Station Hoorn-Kersenboogerd. In: Conference on Computer Assurance, National Institute of Standards and Technology (1995) 57–68
4. Biere, A., Cimatti, A., Zhu, Y.: Symbolic model checking without BDDs. In: TACAS'99. Volume 1579 of LNCS., Springer Verlag (1999)
5. Audemard, G., Bozzano, M., Cimatti, A., Sebastiani, R.: Verifying industrial hybrid systems with MathSAT. In: Bounded Model Checking (BMC'04). Volume 119 of ENTCS. (2005) 17–32
6. Fränzle, M., Herde, C.: HySAT: An efficient proof engine for bounded model checking of hybrid systems. Formal Methods in System Design **30**(3) (2007) 179–198
7. Benhamou, F., Granvilliers, L.: Continuous and interval constraints. In Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Foundations of Artificial Intelligence. Elsevier, Amsterdam (2006) 571–603
8. Barrett, C., Sebastiani, R., Seshia, S., Tinelli, C.: Satisfiability Modulo Theories. In: Handbook on Satisfiability. Volume 185 of Frontiers in Artificial Intelligence and Applications. IO Press (February 2009) ftp://ftp.cs.uiowa.edu/pub/tinelli/papers/BarSST-09.pdf.

9. Bauer, A., Pister, M., Tautschnig, M.: Tool-support for the analysis of hybrid systems and models. In: Design, Automation and Test in Europe, IEEE (2007)
10. Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient Solving of Large Non-linear Arithmetic Constraint Systems with Complex Boolean Structure. JSAT Special Issue on SAT/CP Integration **1** (2007) 209–236
11. Tseitin, G.: On the complexity of derivations in propositional calculus. In: Studies in Constructive Mathematics and Mathematical Logics. (1968)
12. Davis, M., Putnam, H.: A Computing Procedure for Quantification Theory. Journal of the ACM **7**(3) (1960) 201–215
13. Davis, M., Logemann, G., Loveland, D.: A Machine Program for Theorem Proving. Communications of the ACM **5** (1962) 394–397
14. Moore, R.E.: Automatic local coordinate transformation to reduce the growth of error bounds in interval computation of solutions of ordinary differential equations. In Ball, L.B., ed.: Error in Digital Computation. Volume II. Wiley, New York (1965) 103–140
15. Lohner, R.: Einschließung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben. PhD thesis, Fakultät für Mathematik der Universität Karlsruhe, Karlsruhe (1988)
16. Stauning, O.: Automatic Validation of Numerical Solutions. PhD thesis, Technical University of Denmark, Lyngby (1997)
17. Berz, M., Makino, K.: Verified Integration of ODEs and Flows Using Differential Algebraic Methods on High-Order Taylor Models. Reliable Computing **4**(4) (1998) 361–369
18. Henzinger, T.A., Horowitz, B., Majumdar, R., Wong-Toi, H.: Beyond HYTECH: Hybrid systems analysis using interval numerical methods. In Krogh, B., Lynch, N., eds.: Hybrid Systems: Computation and Control. Volume 1790 of LNCS., Springer Verlag (2000) 130–144
19. Hickey, T., Wittenberg, D.: Rigorous modeling of hybrid systems using interval arithmetic constraints. In Alur, R., Pappas, G.J., eds.: Proc. of Hybrid Systems: Computation and Control (HSCC'04). Number 2993 in LNCS, Springer Verlag (2004)
20. Eggers, A., Fränzle, M., Herde, C.: SAT modulo ODE: A direct SAT approach to hybrid systems. In Cha, S.S., Choi, J.Y., Kim, M., Lee, I., Viswanathan, M., eds.: Proceedings of the 6th International Symposiom on Automated Technology for Verification and Analysis (ATVA'08). Volume 5311 of Lecture Notes in Computer Science., Springer (2008) 171–185
21. Fränzle, M., Hermanns, H., Teige, T.: Stochastic Satisfiability Modulo Theory: A Novel Technique for the Analysis of Probabilistic Hybrid Systems. In Egerstedt, M., Mishra, B., eds.: Proceedings of the 11th International Conference on Hybrid Systems: Computation and Control (HSCC'08). Volume 4981 of Lecture Notes in Computer Science., Springer (2008) 172–186
22. Papadimitriou, C.H.: Games against nature. J. Comput. Syst. Sci. **31**(2) (1985) 288–301
23. Schmitt, C.: Bounded Model Checking of Probabilistic Hybrid Automata. Master's thesis, Carl von Ossietzky University, Dpt. of Computing Science, Oldenburg, Germany (March 2008)
24. Teige, T., Fränzle, M.: Stochastic Satisfiability modulo Theories for Non-linear Arithmetic. In Perron, L., Trick, M.A., eds.: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008. Volume 5015 of Lecture Notes in Computer Science., Springer (2008) 248–262

25. Craig, W.: Linear reasoning: A new form of the Herbrand-Gentzen theorem. Journal of Symbolic Logic **22**(3) (1957) 250–268
26. McMillan, K.L.: Interpolation and SAT-based model checking. In Hunt, W.A., Somenzi, F., eds.: CAV. Volume 2725 of Lecture Notes in Computer Science., Springer (2003) 1–13
27. McMillan, K.L.: An interpolating theorem prover. Theor. Comput. Sci. **345**(1) (2005) 101–121
28. Beyer, D., Henzinger, T.A., Jhala, R., Majumdar, R.: The software model checker Blast: Applications to software engineering. International Journal on Software Tools for Technology Transfer (STTT) **9**(5-6) (2007. Invited to special issue of selected papers from FASE 2004/05) 505–525
29. Rybalchenko, A., Sofronie-Stokkermans, V.: Constraint solving for interpolation. In Cook, B., Podelski, A., eds.: VMCAI. Volume 4349 of Lecture Notes in Computer Science., Springer (2007) 346–362
30. Cimatti, A., Griggio, A., Sebastiani, R.: Efficient interpolant generation in satisfiability modulo theories. In: TACAS. (2008)
31. Pudlák, P.: Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations. Journal of Symbolic Logic **62**(3) (September 1997) 981–998
32. Böhm, M., Speckenmeyer, E.: A fast parallel SAT-solver - efficient workload balancing. Annals of Mathematics and Artificial Intelligence **17**(3-4) (1996) 381–400
33. Sinz, C., Blochinger, W., Küchlin, W.: PaSAT - parallel SAT-checking with lemma exchange: Implementation and applications. In Kautz, H., Selman, B., eds.: LICS 2001 Workshop on Theory and Applications of Satisfiability Testing (SAT 2001). Volume 9 of Electronic Notes in Discrete Mathematics., Boston, MA, Elsevier Science Publishers (June 2001)
34. Schubert, T., Lewis, M., Becker, B.: PaMira – A Parallel SAT Solver with Knowledge Sharing. In: 6th International Workshop on Microprocessor Test and Verification (MTV 2005), IEEE Computer Society (2005) 29–36
35. Jurkowiak, B., Li, C.M., Utard, G.: Parallelizing SATZ Using Dynamic Workload Balancing. In: Proceedings of the Workshop on Theory and Applications of Satisfiability Testing (SAT2001). Volume 9., Elsevier Science Publishers (jun 2001)
36. Feldman, Y., Dershowitz, N., Hanna, Z.: Parallel multithreaded satisfiability solver: Design and implementation. Electronic Notes in Theoretical Computer Science **128**(3) (2005) 75–90
37. Lewis, M.D.T., Schubert, T., Becker, B.: Multithreaded SAT solving. In: Proceedings of the 12th Asia and South Pacific Design Automation Conference, IEEE Computer Society (2007) 926–931
38. Zhang, H., Bonacina, M.P., Hsiang, J.: PSATO: A distributed propositional prover and its application to quasigroup problems. Journal of Symbolic Computation **21**(4) (1996) 543–560
39. Ábrahám, E., Schubert, T., Becker, B., Fränzle, M., Herde, C.: Parallel SAT solving in bounded model checking. In Brim, L., Haverkort, B.R., Leucker, M., van de Pol, J., eds.: FMICS/PDMC. Volume 4346 of Lecture Notes in Computer Science., Springer (2006) 301–315
40. Strichman, O.: Accelerating bounded model checking of safety properties. Formal Methods in System Design **24**(1) (2004) 5–24
41. Walsh, T.: Stochastic constraint programming. In: Proc. of the 15th European Conference on Artificial Intelligence (ECAI'02), IOS Press (2002)