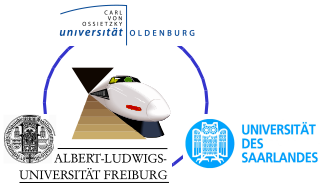

AVACS – Automatic Verification and Analysis of Complex Systems

REPORTS

of SFB/TR 14 AVACS

Editors: Board of SFB/TR 14 AVACS

SAT Modulo ODE: A Direct SAT Approach to Hybrid Systems

by
Andreas Eggers Martin Fränzle Christian Herde

AVACS Technical Report No. 37
April 2008
ISSN: 1860-9821

Publisher: Sonderforschungsbereich/Transregio 14 AVACS
(Automatic Verification and Analysis of Complex Systems)
Editors: Bernd Becker, Werner Damm, Martin Fränzle, Ernst-Rüdiger Olderog,
Andreas Podelski, Reinhard Wilhelm
ATRs (AVACS Technical Reports) are freely downloadable from `www.avacs.org`

Author(s) contact: Martin Fänzle (`mf@avacs.org`).

SAT Modulo ODE: A Direct SAT Approach to Hybrid Systems

Andreas Eggers, Martin Fränzle, and Christian Herde*

Dept. of CS, Carl von Ossietzky Universität Oldenburg, Germany
{eggers|fraenzle|herde}@informatik.uni-oldenburg.de

Abstract. In order to facilitate automated reasoning about large Boolean combinations of non-linear arithmetic constraints involving ordinary differential equations (ODEs), we provide a seamless integration of safe numeric overapproximation of initial-value problems into a SAT-modulo-theory (SMT) approach to interval-based arithmetic constraint solving. Interval-based safe numeric approximation of ODEs is used as an interval contractor being able to narrow candidate sets in phase space in both temporal directions: post-images of ODEs (i.e., sets of states reachable from a set of initial values) are narrowed based on partial information about the initial values and, vice versa, pre-images are narrowed based on partial knowledge about post-sets.
In contrast to the related CLP(F) approach of Hickey and Wittenberg [10], we do (a) support coordinate transformations mitigating the wrapping effect encountered upon iterating interval-based overapproximations of reachable state sets and (b) embed the approach into an SMT framework, thus accelerating the solving process through the algorithmic enhacements of recent SAT solving technology.

1 Introduction

Hybrid systems consist of interacting discrete and continuous components, with the continuous components often being naturally described by a combination of ordinary differential equations (ODEs), formalizing time-dependent continuous behavior, and arithmetic (in-)equations portraying autonomous jumps, invariants, and the like. Automating state-exploratory analysis of hybrid systems does thus call for effective manipulation of Boolean combinations of the above, where the large discrete state spaces encountered in real systems and the dependence of the continuous behavior on the current discrete state gives rise to extremely large Boolean combinations. Within this paper, we suggest a SAT modulo theory (SMT) approach for directly handling these large compositions of ODEs, arithmetic (in-)equations, and conditions on discrete states.

* This work has been partially funded 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).

Our approach draws from three up to now distinct technologies, trying to combine their virtues: (1) Solving large and complex-structured Boolean combinations of arithmetic constraints by *SAT modulo theory* techniques (e.g., [8, 5] These approaches are attractive as they transfer the algorithmic enhancements that were instrumental to the enormous performance gains recently achieved in propositional SAT solving, like non-chronological backtracking and conflict-driven learning, to the mixed Boolean-arithmetic domain, as encountered in hybrid systems [1]. (2) *Interval-based safe numeric approximation of ODEs*, as suggested by, a.o., Moore, Lohner, and Stauning [12, 11, 14]. This approach provides a technique for safely overapproximating the image under an ODE of a rectangular region in phase space and incorporates techniques based on coordinate transformations for mitigating the wrapping effect encountered upon iterating interval-based overapproximations of reachable state sets. (3) *CLP(F)* [10], offering a symbolic, constraint-based technology for reasoning about ODEs grounded in (in-)equational constraints obtained from Taylor expansions, thus being able to handle ODE parameters, error ranges in measurements, and other natural uncertainties in modeling dynamic systems. Such effects are hard to deal with, and hence often ignored, within numeric approaches to image computation.

Our design goal was to resolve the following shortcomings of the aforementioned techniques: First, the SMT framework, while being able to handle very large constraint systems involving arithmetic (in-)equations, did previously lack any native support for ODEs. Second, CLP(F) may fail to provide tight approximations of the ODE solutions due to not counterfeiting the *wrapping effect* [12] encountered in iterating interval-based, i.e. rectangular, overapproximations of state sets. Furthermore, CLP(F) lacks the sophisticated means of pruning the search space based on conflict analysis found in recent SMT solvers.[1] Third, the tighter approximations computed by interval-based safe numeric approximation of ODEs, which have successfully been used in state-exploratory verification of hybrid systems (e.g., in Hypertech [9]), lack the constraint propagation and reasoning functionality of the CLP(F) approach. Instead, their use was confined to extrapolating state sets in an a priori fixed temporal direction of exploration.

To mitigate these restrictions, we suggest a direct, seamless integration of safe approximation of ODEs into the iSAT arithmetic constraint solver [7], which is an adaptation of the SMT framework to the undecidable domain of non-linear arithmetic involving, a.o., inequations entailing transcendental functions. On the theory solver side, iSAT is based on interval constraint propagation (ICP) for arithmetic (in-)equations [2], which we extend to ODEs as follows. Interval-based safe numeric approximation of ODEs is used as an interval contractor being able to narrow candidate sets in phase space in both temporal directions: post-images of ODEs (i.e., sets of states reachable from a set of initial values) are narrowed based on partial information about the initial values and, vice versa, pre-images are narrowed based on partial knowledge about post-sets.

[1] Due to the generality of the CLP framework, the programmer may nevertheless be able to simulate many of these pruning operators, albeit at the price of a very imperative use of CLP.

Structure of the paper. Section 2 explains syntax and semantics of the arithmetic satisfiability problems we are going to address. In Sect. 3, we then start our exposition of the solver algorithms with a description of the arithmetic SAT solving technology we build upon. Thereafter, we provide a detailed exposition of its extension to ODEs (Sect. 4) and benchmark results indicating feasibility of the technique (Sect. 5).

2 Arithmetic SAT problems involving ODEs

Aiming at automated analysis of hybrid systems, our constraint solver addresses satisfiability of non-linear arithmetic constraints, including ODEs, over real-valued variables RV plus Boolean variables BV for encoding the control flow. The user thus may input constraint formulae built from quantifier-free (in-)equational constraints over the reals, from ODEs, and from propositional variables using arbitrary Boolean connectives. The atomic (in-)equational constraints are relations between potentially non-linear terms involving transcendental functions, like $\sin(x+\omega t)+ye^{-t} \leq z+5$. The ODE constraints define the derivatives of the continuous variables w.r.t. time. They are given by equational constraints of the form $\frac{dx_i(t)}{dt} = f_i(\vec{x}(t))$, where the ODE-defined variables x_i constitute a vector $\vec{V}$ over a subset of RV and f_i are potentially non-linear expressions over $\vec{V}$. Additionally *flow invariants* of the form $l \leq x_i \leq u$ can be given that constrain the range of the variables in V during a continuous flow.[2]

An input model comprises predicative encodings of the initial state set *init*, the transition relation *trans* over current-step (x) and next-step variables (x'), and the (unsafe) *target* state. ODE constraints can only occur in the transition relation where they define the relationship between two successive valuations of the variables in V by constraining the possible trajectories in between the steps. In order to perform bounded model checking (BMC) [3] on such model encodings, the transition relation is unwound k times and conjoined with the predicates that encode initial and target states, yielding a formula

$$\phi = init(\vec{x}^{(0)}) \wedge trans(\vec{x}^{(0)}, \vec{x}^{(1)}) \wedge \cdots \wedge trans(\vec{x}^{(k-1)}, \vec{x}^{(k)}) \wedge target(\vec{x}^{(k)}) \quad (1)$$

that is satisfiable iff a state satisfying *target* is reachable in k steps. Each variable x_i thus results in $k+1$ instances $x_i^{(0)}, x_i^{(1)}, \ldots, x_i^{(k)}$. If ϕ is satisfied by the valuations of all instances for all variables occuring in ϕ, these valuations represent the evolution of the system during a particular *trace*. Like all other subexpressions of the transition system, also the ODE constraints are instantiated k times. The resulting formula ϕ thus contains ODE constraints over disjoint sets of variable instances $V^{(0)}, \ldots, V^{(k)}$, where each $V^{(i)}$ contains those instances whose ODE-defined trajectories emerge from the valuations in the i-th step.

As the ODEs that govern the hybrid-system behavior depend on the current discrete state of the system, such ODE constraints are relativized by the

[2] We define a fixed ordering over the variables from $RV \cup BV$ s.t. we can exchange the vector $\vec{V} = (x_1, \ldots, x_n)^T$ of unique variables with the set $V = \{x_1, \ldots, x_n\} \subseteq RV$ and vice versa.

introduction of propositional "triggers", such that traces need only obey the pre-/post-relation defined by those ODE constraints and the flow invariants whose triggers are forced to true by predicates formalizing the interplay between discrete state and continuous behavior. Several of these trigger variables together can in principle activate ODE constraints that describe conflicting dynamics for the same unwinding depth of the transition relation. To avoid this, constraints can be added such that only one type of dynamics may be active at a time by a simple investigation of the variables that are influenced by the triggers.

Solver-internal constraint syntax. By the front-end of our solver, constraint formulae are rewritten into equi-satisfiable quantifier-free formulae in conjunctive normal form, with atomic propositions ranging over propositional variables and (in-)equational constraints confined to a form resembling three-address code. This rewriting is based on the standard mechanism of introducing auxiliary variables for the values of arithmetic sub-expressions and of logical sub-formulae, thereby eliminating common sub-expressions and sub-formulae through re-use of the auxiliary variables, thus reducing the search space of the SAT solver and enhancing the reasoning power of the interval contractors used in arithmetic reasoning [2]. Thus, the *internal* syntax of constraint formulae is as follows:

$$
\begin{aligned}
\mathit{formula} &::= \{\mathit{clause} \wedge\}^* \mathit{clause} \\
\mathit{clause} &::= (\{\mathit{atom} \vee\}^* \mathit{atom}) \mid (\mathit{bound} \Rightarrow \mathit{ode} \wedge \mathit{flow\_invar}) \\
\mathit{atom} &::= \mathit{bound} \mid \mathit{equation} \\
\mathit{bound} &::= \mathit{variable} \sim \mathit{rational\_constant} \\
\mathit{variable} &::= \mathit{real\_var} \mid \mathit{boolean\_var} \\
\mathit{equation} &::= \mathit{real\_var} = \mathit{real\_var}\ \mathit{bop}\ \mathit{real\_var} \mid \mathit{real\_var} = \mathit{uop}\ \mathit{real\_var} \\
\mathit{ode} &::= \{\tfrac{\mathrm{d}\mathit{real\_var}}{\mathrm{d}t} = \mathit{term} \wedge\}^* \tfrac{\mathrm{d}\mathit{real\_var}}{\mathrm{d}t} = \mathit{term} \\
\mathit{flow\_invar} &::= \{\mathit{bound} \wedge\}^* \mathit{bound}
\end{aligned}
$$

where $\sim \in \{<, \leq, >, \geq\}$, the non-terminals *bop*, *uop* denote the binary and unary operator symbols (including arithmetic operators such as + or sin), and *term* the terms over real-valued variables built using these.

Semantics. Such constraint formulae are interpreted over valuations $\sigma \in (BV \xrightarrow{\text{total}} \mathbb{B}) \times (RV \xrightarrow{\text{total}} \mathbb{R})$, where BV is the set of Boolean and RV the set of real-valued variables, being the instances of the variables that result from the BMC unwinding depicted in (1). $\mathbb{B}$ is identified with the subset $\{0, 1\}$ of $\mathbb{R}$ such that bounds on a Boolean variable v correspond to literals v or $\neg v$. The definition of satisfaction is standard: a constraint formula ϕ is satisfied by a valuation iff all its clauses are satisfied. A disjunctive clause is satisfied iff at least one of its atoms is satisfied. Satisfaction of atoms is wrt. the standard interpretation of the arithmetic operators and the ordering relations over the reals. We assume all arithmetic operators to be total and therefore extend their codomain (as well as, for compositionality, their domain) with a special value $\mho \notin \mathbb{R}$ ("undefined") such that the operators manipulate values in $\mathbb{R}^{\mho} = \mathbb{R} \cup \{\mho\}$. The comparison operations on $\mathbb{R}$ are extended to $\mathbb{R}^{\mho}$ in such a way that $\mho$ is incomparable to any real number, that is, $c \not\sim \mho$ and $\mho \not\sim c$ for any $c \in \mathbb{R}$ and any relation $\sim \in \{<, \leq, =, \geq, >\}$. ODE constraints are satisfied if there exists for each BMC

unwinding depth i a solution of the ODE system $\frac{d\vec{x}(t)}{dt} = \vec{f}(\vec{x}(t))$ where the activated triggers of that BMC depth i define which ODE constraints are used as components $f_1, \ldots, f_n$. Such a solution function $\vec{x}(t)$ satisfies the ODE up to a user-specified *horizon* of interest, for its starting point $\vec{x}(0) = \sigma(\vec{x}^{(i)})$ holds, i.e. the trajectory emerges from the current valuation of $\vec{x}$ on BMC depth i, and there exists a $\tau_r \in [0, horizon]$ such that $\vec{x}(\tau_r) = \sigma(\vec{x}^{(i+1)})$, i.e. the trajectory eventually reaches the next value of $\vec{x}$ in the trace.

Interval-based overapproximation. Instead of real-valued valuations of variables, our constraint solving algorithm manipulates interval-valued valuations $\rho \in (BV \xrightarrow{\text{total}} \mathbb{I}_\mathbb{B}) \times (RV \xrightarrow{\text{total}} \mathbb{I}_\mathbb{R})$, where $\mathbb{I}_\mathbb{B} = 2^\mathbb{B}$ and $\mathbb{I}_\mathbb{R}$ is the set of convex subsets of $\mathbb{R}^{\mho}$.[3] Slightly abusing notation, we write $\rho(l)$ for $\rho_{\mathbb{I}_\mathbb{B}}(l)$ when $\rho = (\rho_{\mathbb{I}_\mathbb{B}}, \rho_{\mathbb{I}_\mathbb{R}})$ and $l \in BV$, and similarly $\rho(x)$ for $\rho_{\mathbb{I}_\mathbb{R}}(x)$ when $x \in RV$. For a vector of ODE-defined variables $\vec{V} = (x_1, \ldots, x_n)^T$, we write $\rho(\vec{V})$ as an abbreviation for the vector $(\rho(x_1), \ldots, \rho(x_n))^T$. In the following, we occasionally use the term *box* synonymously for interval valuation. If both ζ and η are interval valuations then ζ is called a *refinement* of η iff $\zeta(v) \subseteq \eta(v)$ for each $v \in BV \cup RV$.

In order to lift a binary operation $\circ$ and its partial inverses to sets, we define

$$m \bullet_1 n = \{x \mid \exists y \in m, z \in n : x = y \circ z\},$$
$$m \bullet_2 n = \{y \mid \exists x \in m, z \in n : ((x = y \circ z) \vee (y = \mho \wedge \nexists y' \in \mathbb{R} : x = y' \circ z))\}$$
$$m \bullet_3 n = \{z \mid \exists x \in m, y \in n : ((x = y \circ z) \vee (z = \mho \wedge \nexists z' \in \mathbb{R} : x = y \circ z'))\}$$

and similarly for unary $\circ$. Note that these are essentially the images of the argument sets under the relation $\{(x, y, z) \mid x = y \circ z\}$ when substituting the respective arguments.

As these operations can in general not be carried out exactly using floating-point arithmetic, we lift the set-valued operators to (computer-representable) intervals by assigning to each set-valued operation $\bullet$ a conservative interval approximation $\hat{\bullet}$ which satisfies $i_1 \hat{\bullet} i_2 \in \mathbb{I}_\mathbb{R}$ and $i_1 \hat{\bullet} i_2 \supseteq i_i \bullet i_2$ for all intervals i_1 and i_2 [13]. Note that the definition of an interval extension does not specify how to exactly lift a base operation $\bullet$ to intervals, but leaves some design choice by permitting arbitrary overapproximations. For the sake of reasoning power, $i_1 \hat{\bullet} i_2$ should be chosen such that it provides an as tight as possible overapproximation of $i_1 \bullet i_2$. This means that in practice $i_1 \hat{\bullet} i_2$ is the *interval hull* $\bigcap_{i \in \mathbb{I}_\mathbb{R}, i \supseteq i_1 \bullet i_2} i$ — i.e. the smallest interval in $\mathbb{I}_\mathbb{R}$ that contains $i_1 \bullet i_2$ entirely — extended by some outward rounding to compensate for the imprecision of computer arithmetic and the finiteness of the set of floating-point numbers. We define *narrowing operators* $\pi^{\hat{\bullet}}$ for each of the relations introduced above. For a constraint $x = y \circ z$, we define the operators $\pi_1^{\hat{\bullet}}(\rho(x)) := \rho(x) \cap (\rho(y) \hat{\bullet}_1 \rho(z))$, $\pi_2^{\hat{\bullet}}(\rho(y)) := \rho(y) \cap (\rho(x) \hat{\bullet}_2 \rho(z))$, and $\pi_3^{\hat{\bullet}}(\rho(z)) := \rho(z) \cap (\rho(x) \hat{\bullet}_3 \rho(y))$. The essential characteristics of these narrowing operators, which are borrowed from the context of hull consistency [2], is that they narrow the valuation ρ by pruning away only parts of the search space that cannot contain any satisfying valuations.

[3] Note that this definition covers the open, half-open, and closed intervals over $\mathbb{R}$, including unbounded intervals, as well as the union of such intervals with $\{\mho\}$.

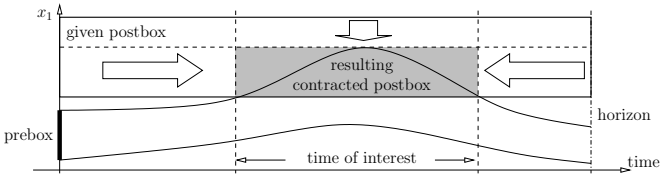

Fig. 1. Find trajectories that emerge from the prebox and eventually reach the postbox

In order to extend the concept of interval-based narrowing operators to ODEs, we first introduce the following definitions. We consider an *ODE problem*

$$\mathcal{P} := \left(\tfrac{d\vec{x}}{dt}(t) = \vec{f}(\vec{x}(t)), \vec{X}_{pre}, \vec{X}_{post}, \vec{X}_{flow}\right) \tag{2}$$

where $\frac{d\vec{x}}{dt}(t) = \vec{f}(\vec{x}(t))$ is an n-dimensional system of time-invariant differential equations with components $\frac{dx_i}{dt}(t) = f_i(x_1(t), \ldots, x_n(t))$, with $i \in \{1, \ldots, n\}$ and $\vec{X}_{pre}$, $\vec{X}_{post}$, and $\vec{X}_{flow}$ are the *prebox*, *postbox*, and *flowbox* respectively, which are vectors of real-valued intervals for the variables $x_1, \ldots, x_n$. The flowbox is defined by the conjunction of activated flow constraints for the BMC depth i to which $\mathcal{P}$ belongs. The prebox is given by $\vec{X}_{pre} = \rho((x_1^{(i)}, \ldots, x_n^{(i)})^T)$ and the postbox by the corresponding next values $\vec{X}_{post} = \rho((x_1^{(i+1)}, \ldots, x_n^{(i+1)})^T)$. This relationship is illustrated in Fig. 1.

Similarly to $\pi^{\hat{\bullet}}$ for equational atoms of ϕ, we define narrowing operators for the ODE problem $\mathcal{P}$ that results from the active triggers on BMC depth i:

$$\begin{aligned}
&\pi^{\mathcal{P}}_{\rightarrow}(\vec{X}_{post}) := \vec{X}_{post} \ \cap \\
&\quad \varepsilon\Big(\Big\{\vec{y} \;\Big|\; \exists \tau_r \in [0, horizon], \exists \vec{x} : [0, \tau_r] \to \mathbb{R}^n : \vec{x}(0) \in \vec{X}_{pre} \ \ {}^{//\vec{x}(t) \text{ emerges}}_{\text{from prebox}} \\
&\qquad \wedge \ \forall \tau \in [0, \tau_r] : \frac{d\vec{x}}{dt}(\tau) = \vec{f}(\vec{x}(\tau)) \quad {}^{//\text{is a solution}} \\
&\qquad \wedge \ \vec{y} = \vec{x}(\tau_r) \ \wedge \ \forall \tau \in [0, \tau_r] : \vec{x}(\tau) \in \vec{X}_{flow}\Big\}\Big) \ {}^{//\text{eventually reaches } \vec{y} \text{ without leaving the flowbox}}
\end{aligned}$$

and similarly for the inverse direction

$$\begin{aligned}
&\pi^{\mathcal{P}}_{\leftarrow}(\vec{X}_{pre}) := \vec{X}_{pre} \ \cap \\
&\quad \varepsilon\Big(\Big\{\vec{y} \;\Big|\; \exists \tau_r \in [0, horizon], \exists \vec{x} : [0, \tau_r] \to \mathbb{R}^n : \vec{x}(0) = \vec{y} \quad {}^{//\vec{x}(t) \text{ emerges}}_{\text{from } \vec{y}} \\
&\qquad \wedge \ \forall \tau \in [0, \tau_r] : \frac{d\vec{x}}{dt}(\tau) = \vec{f}(\vec{x}(\tau)) \quad {}^{//\text{is a solution}} \\
&\qquad \wedge \ \vec{x}(\tau_r) \in \vec{X}_{post} \ \wedge \ \forall \tau \in [0, \tau_r] : \vec{x}(\tau) \in \vec{X}_{flow}\Big\}\Big) \ {}^{//\text{reaches postbox without leaving flowbox}}
\end{aligned}$$

where ε is an overapproximating interval enclosure of its argument. As all valuations that are reachable from the prebox are enclosed by the narrowed postbox and all starting points of trajectories that can eventually reach the postbox are enclosed by the narrowed prebox, no solution of $\mathcal{P}$ can be lost by applying the operators $\pi^{\mathcal{P}}_{\rightarrow}$ and $\pi^{\mathcal{P}}_{\leftarrow}$. This mostly declarative description of a pruning operator

for ODEs, that does only remove non-solutions from the pre- and postbox, is complemented by a presentation of its practical implementation in Sect. 4.

3 Arithmetic constraint solving with the iSAT algorithm

The approach to hybrid systems verification pursued in this paper is to translate the reachability problem of a hybrid system into a satisfiability problem of a constraint system. The BMC formula (1) in the form of disjunctive clauses over Boolean literals, arithmetic constraints, and ODEs is automatically generated from the model description and then becomes input for the satisfiability check.

For a purely discrete transition system, the BMC formula ϕ would be propositional. The common approach to solve such SAT problems in the Boolean domain is the DPLL [4] procedure that performs a (often non-chronological) backtrack search through the solution space by alternating decision and deduction steps. Decisions are assignments of `true` or `false` to a still undecided variable; deductions consist of finding atoms that have become the last remaining free atom of a clause in which all other atoms have become `false`. These atoms are satisfied by assigning them to `true`, called a *unit propagation* (UP). Their new valuations are subsequently propagated to all the occurrences of the respective literals, potentially triggering further UPs. When all UPs have been performed and the formula still has no clause entirely assigned to `false`, a decision is conducted, selecting one of the free variables. As soon as a clause becomes unsatisfiable by all its atoms being `false`, a decision and its implied UPs are undone and the search thus continued from a higher level in the search tree.

In [7], the iSAT algorithm has been proposed to lift DPLL to the domain of non-linear arithmetic problems with complex Boolean structure. The basic additions to DPLL are, first, that variables are no longer interpreted by Boolean intervals in $\mathbb{I}_\mathbb{B}$ only, but — depending on type — also by real-valued intervals in $\mathbb{I}_\mathbb{R}$ and, second, that interval constraint propagation (ICP) by the narrowing operators for equations (cf. previous section) joins UP as a deduction mechanism.

Initially, all variables are interpreted by intervals coinciding to their specified ranges, e.g. $[-133, 450]$ for a real-valued variable of that range and $\{\texttt{false}, \texttt{true}\}$ for a Boolean variable. To simplify the exposition, we do also consider all atoms occurring in the formula to be Boolean variables. As in DPLL, progress in constructing the proof tree is now made by applying decision and deduction steps. Akin to DPLL, decisions are simply taken by splitting the interval which bounds the current valuation of a variable x into two disjoint, non-empty subintervals (e.g. at the midpoint) and recursing the search on one of them. Deduction is more general than in DPLL, applying the following forms of propagation rules:

1. As in DPLL, unit propagation sets the last remaining atom in a clause to `true` if all others have been assigned `false`. For the sake of efficiency, watch lists are applied for detecting this.
2. If an equational atom $x = y \circ z$ or $x = \circ y$ has been assigned `true`, its associated narrowing operators $\pi_i^\bullet$ can be used to chop off valuations violating the equational constraint, thus narrowing the intervals interpreting x, y, and z.

For efficiency, this interval constraint propagation is triggered by watch lists observing whether one of the variables of an equational atom has changed.
3. If a bound atom $x \sim c$ becomes assigned `true`, which is detected by means of the watch lists, the domain of x is intersected with $\{u \in \mathbb{R} \mid u \sim c\}$.
4. If an equational or bound atom becomes inconsistent in the sense that application of its associated narrowing rules would yield an empty interval for some variable, the atom is assigned `false` if it has not been assigned `true` previously. In the latter case, a *conflict* arises which manifests itself by one of the ICP propagation rules 2. and 3. assigning an empty interval.

As in DPLL, preference over decisions is given to deduction. Exploiting the similarity of this algorithm to DPLL by incorporating and generalizing the conflict analysis and conflict-driven learning algorithms of modern DPLL-based SAT solvers, together with non-chronological backtracking, this yields a solver for large and complex-structured mixed Boolean-arithmetic constraint systems involving non-linear and even transcendental arithmetic operations [7]. Applying deductions and decisions until a sufficiently small consistent box has been found or absence thereof has been proved, it decides constraint formulae provided they are robust in the sense that some open set of solutions exists, if any.

By integrating the narrowing operators described in the following section, we extend this algorithm to check the consistency of and to deduce new bounds from ODE constraints, thereby directly handling them within the BMC formula.

4 Contracting pre- and postimages of ODE trajectories

In order to handle differential equations directly, safe overapproximations of their solution sets must be generated during the deduction phase. We consider an ODE Problem $\mathcal{P}$ of the form (2). As motivated in Sect. 2, the goal of calling the ODE solver is to enclose all trajectories of $\mathcal{P}$ that are (sufficiently often) differentiable solution functions $\vec{x}$, emerging from the prebox, eventually reaching the postbox, and staying in the flowbox during their evolution.

These trajectories then allow to define a contraction $\vec{X'}_{post} \subseteq \vec{X}_{post}$ that constitutes new bounds on the variables of the postbox which can subsequently be propagated through the other constraints and thus cause further deductions. The contracted set $\vec{X'}_{post}$ must contain all points that are reachable by the trajectories and are included in the given $\vec{X}_{post}$. The postbox thus defines the set of points which are interesting to the surrounding deduction. We call the set of those points of time for which trajectories exist that have a valuation inside $\vec{X}_{post}$ (and thereby also in $\vec{X'}_{post}$) the *time of interest* (ToI).

By multiplying the right hand side of the ODE, i.e. $\vec{f}(\vec{x}(t))$, with -1 and using $\vec{X}_{post}$ in place of $\vec{X}_{pre}$ and vice versa, the inverse problem

$$\mathcal{P}^{-1} := \left(\tfrac{d\vec{x}}{dt} = -\vec{f}(\vec{x}(t)), \vec{X}_{post}, \vec{X}_{pre}, \vec{X}_{flow}\right)$$

is generated. $\mathcal{P}^{-1}$ then allows to also contract $\vec{X}_{pre}$ into $\vec{X'}_{pre} \subseteq \vec{X}_{pre}$ by propagating the postbox backwards through the ODE problem using the same ToI.

Enclosure mechanism. In order to enclose these trajectories, we first generate by means of symbolic derivation and simplification rules the truncated Taylor series of the unknown exact solution up to the user-specified order m and the corresponding Lagrange remainder term of order $m+1$ that will be used to enclose all possible truncation errors. These symbolic terms need only to be generated once for all dimensions $i \in 1, \ldots, n$ of the ODE problem $\mathcal{P}$:

$$x_i(t_k + h_k) = \underbrace{\sum_{j=0}^{m} \frac{h_k^j}{j!} \frac{d^j x_i}{dt^j}(t_k)}_{\text{Truncated Taylor series}} + \underbrace{\frac{h_k^{m+1}}{(m+1)!} \frac{d^{m+1} x_i}{dt^{m+1}}(t_k + \theta h_k)}_{\text{Lagrange remainder with } \theta \in]0,1[}$$

Essentially following the approach of Lohner [11], each enclosure step from t_k to t_{k+1} consists of two tasks: First, we generate a rough overapproximation of the trajectories ("bounding box") along with a suitable stepsize h_k for which the bounding box guarantees to enclose all trajectories. Second, we evaluate the Taylor series over the calculated box at t_k (with the box at t_0 being the given prebox) and the stepsize h_k and calculate interval bounds for the Lagrange remainder over the bounding box using outward rounding for interval calculations as described in Sect. 2. The first step, i.e. finding a rough a-priori enclosure of the solution set over the interval of time from $[t_k, t_k + h_k]$ is based on a theorem given by Lohner [11, p. 263]. Extending the Picard-Lindelöf existence and uniqueness theorem for initial value problems, it allows to easily decide whether a given box encloses the trajectories emerging from a box $\vec{X}_k$ over $[t_k, t_k + h_k]$. We use this property in a greedy search algorithm that (starting from the local starting box $\vec{X}_k$) extends the box into one direction at a time and checks whether the stepsize for which this box guarantees to be a bounding box has grown.

We call the vector of truncated Taylor series $\vec{TT}(\vec{X}_k, h_k) =$

$$\begin{pmatrix} \sum_{j=0}^{m} \frac{h_k^j}{j!} \frac{d^{j-1} f_1}{dt^{j-1}}(\vec{X}_k) \\ \vdots \\ \sum_{j=0}^{m} \frac{h_k^j}{j!} \frac{d^{j-1} f_n}{dt^{j-1}}(\vec{X}_k) \end{pmatrix} \supseteq \begin{pmatrix} \sum_{j=0}^{m} \frac{h_k^j}{j!} \frac{d^{j-1} f_1}{dt^{j-1}}(\vec{x}(t_k)) \\ \vdots \\ \sum_{j=0}^{m} \frac{h_k^j}{j!} \frac{d^{j-1} f_n}{dt^{j-1}}(\vec{x}(t_k)) \end{pmatrix} = \begin{pmatrix} \sum_{j=0}^{m} \frac{h_k^j}{j!} \frac{d^j x_1}{dt^j}(t_k) \\ \vdots \\ \sum_{j=0}^{m} \frac{h_k^j}{j!} \frac{d^j x_n}{dt^j}(t_k) \end{pmatrix}$$

with $\vec{X}_k \supseteq \vec{x}(t_k)$ being an overapproximating enclosure of the exact solution set at t_k. The first enclosure at $t_0 = 0$ is given by the prebox: $\vec{X}_0 = \vec{X}_{pre} = \vec{x}(t_0)$. Similarly we call the vector of the error enclosure terms $\vec{EE}(\vec{BB}_k, h_k) =$

$$\begin{pmatrix} \frac{h_k^{m+1}}{(m+1)!} \frac{d^m f_1}{dt^m}(\vec{BB}_k) \\ \vdots \\ \frac{h_k^{m+1}}{(m+1)!} \frac{d^m f_n}{dt^m}(\vec{BB}_k) \end{pmatrix} \supseteq \begin{pmatrix} \frac{h_k^{m+1}}{(m+1)!} \frac{d^{m+1} x_1}{dt^{m+1}}(t_k + \theta h_k) \\ \vdots \\ \frac{h_k^{m+1}}{(m+1)!} \frac{d^{m+1} x_n}{dt^{m+1}}(t_k + \theta h_k) \end{pmatrix}$$

where $\vec{BB}_k \supseteq \vec{x}([t_k, t_k + h_k])$ is a bounding box that safely overapproximates all trajectories over the interval $[t_k, t_k + h_k]$.

Calculating the interval overapproximation of the symbolically given truncated Taylor term and adding the safely enclosed error remainder yields a box

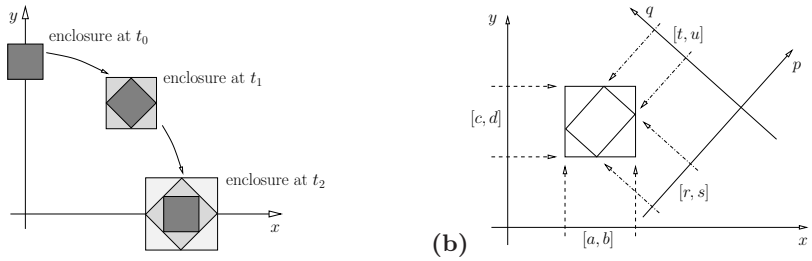

Fig. 2. (a) wrapping effect, (b) coordinate transformation (origin shifted for clarity)

that encloses all trajectories at the next point of time $t_{k+1} = t_k + h_k$:

$$\vec{X}_{k+1}^{\text{naive}} \supseteq \vec{TT}(\vec{X}_k, h_k) + \vec{EE}(\vec{BB}_k, h_k) \tag{3}$$

This "naive" enclosure could be iterated until the given horizon is reached. If instead of (3) we evaluated

$$\vec{X}_{[k,k+1]}^{\text{naive}} \supseteq \vec{TT}(\vec{X}_k, [0, h_k]) + \vec{EE}(\vec{BB}_k, [0, h_k])$$

and generated the union

$$\vec{X}_{post}^{naive} = \bigcup_{k \in \{0,\ldots,q\}} \vec{X}_{[k,k+1]}^{\text{naive}}, \text{ with } t_q \geq horizon$$

we would actually receive a correct enclosure of all possible trajectories emerging from $\vec{X}_{pre}$ over $[t_0, t_q]$. However, a fundamental problem — the so called *wrapping effect* — arises that renders these enclosures useless in most cases.

Consider the example of a harmonic oscillator that is used by Moore [12] to illustrate the wrapping effect: $\frac{dx}{dt} = y \ \wedge \ \frac{dy}{dt} = -x$. Manually calculating the exact solutions for a box of initial values yields the behaviour depicted in Fig. 2. The solution sets at the shown t_k expose a rotation without any change in the size of the solutions. Using the naive enclosure mechanism, we do however have to enclose the rotated solution sets with boxes that are parallel to the coordinate axes and thereby wrap in points that are not part of the solution. By iterating this method we consequently have to calculate enclosures for all those trajectories emerging from these wrapped points. This leads to an exponential blow up of the enclosures thus causing often inacceptably coarse overapproximations [12].

The standard approach to mitigate this problem is given already by Moore: In order to keep the enclosure tight, the coordinate system is changed in such a way that it minimizes the wrapping effect, i.e. the coordinate system with respect to which the enclosure boxes are given is rotated (and even sheared) along with the solution set and thereby allows to enclose the solutions much more tightly.

By orthogonalizing the transformation matrix used to conduct the coordinate transformation (QR method), Lohner successfully avoids this matrix to become singular and therefore the transformation to fail [11]. In our prototypical implementation, a simple threshold on the minimum angle of the coordinate axes is used to decide whether that QR conditioning is applied or not.

We modify the naive method sketched above by first creating a well-suited coordinate system for the next step at t_{k+1}. In order to achieve this, we take the midpoints of the surfaces defined by $\vec{X}_k$ and calculate approximations of their image points at t_{k+1}. The connections between the resulting points are an approximation of the optimal coordinate system for an enclosure at t_{k+1}. The enclosure for t_0 is given wrt. the standard coordinate system. Coordinate transformations are achieved by performing matrix multiplication with a transformation matrix T whose columns are the images of the standard base under the coordinate transformation and with its inverse T^{-1}, depending on the direction of the transformation. In order to protect the overapproximation guarantees, we need to ensure that $I \in T \cdot T^{-1}$, where I is the unit matrix. We therefore iteratively extend an approximative T^{-1} until $I \in T \cdot T^{-1}$ holds.

Instead of interval enclosures $\vec{X}_k$ with respect to the standard base we store interval enclosures $\vec{U}_k$ with respect to the coordinate system defined by the transformation matrices T_k and its safely enclosed inverse T_k^{-1}. In order to yield a wrapped enclosure at t_k, one can evaluate $\vec{X}_k = T_k \cdot \vec{U}_k$ with safe interval extended operators. As we have symbolic representations of $\vec{TT}$ and $\vec{EE}$, we can avoid this internal wrapping during evaluation by symbolically simplifying

$$\vec{U}_{k+1} = T_{k+1}^{-1} \cdot \begin{pmatrix} TT_1(T_k \cdot \vec{U}_k, h_k) \\ \vdots \\ TT_n(T_k \cdot \vec{U}_k, h_k) \end{pmatrix} + T_{k+1}^{-1} \cdot \begin{pmatrix} EE_1(\vec{BB}_k, h_k) \\ \vdots \\ EE_n(\vec{BB}_k, h_k) \end{pmatrix}$$

prior to its evaluation. By intersecting the intermediate results $\vec{X}_k$ and the bounding boxes with $\vec{X}_{flow}$, we prune off some of those trajectories that are no longer interesting after the k-th step because they then left the flowbox, we currently do however not use a coordinate-transformed version of the flowbox to also prune the boxes $\vec{U}_k$ that are used for iteration.We can stop iterating the method before reaching the horizon when $\vec{X}_k \cap \vec{X}_{flow}$ becomes empty.

Again, replacing h_k with $[0, h_k]$ and subsequent generation of the union of the local enclosures from t_0 to t_q leads to an enclosure of all trajectories over $[t_0, t_q]$. By locally intersecting this enclosure with $\vec{X}_{post}$, the tightened postbox $\vec{X'}_{post}$ can be generated. Enclosing all subintervals of time with non-empty intersections of $\vec{X}_k$ and $\vec{X}_{post}$, yields an overapproximation of the time of interest.

As seen above, this method can generate almost optimal enclosures for solution sets that are affine images of the prebox. Lohner points out that we cannot expect this method to work for nonlinear ODEs as good as it works in the case of linear ODEs [11], whose solution sets are always given by affine transformations. Though the method has no fundamental restriction to linear ODEs, coordinate transformations of the described flavor are in general only effective in the case of linear ODEs. The coarseness of the enclosures of nonlinear ODEs thus strongly depends on whether the ODE itself causes a contraction of the solution sets that is stronger than the expansion caused by the wrapping effect.

Integration into the iSAT algorithm. In contrast to arithmetic constraints, the method described above to perform enclosures of ODE trajectories requires a

fully defined ODE problem that normally consists of more than one ODE constraints. These are activated by trigger variables as described in Sect. 2. Prior to executing the enclosure algorithm it is thus necessary to collect the activated ODE constraints and construct the ODE problems they define. During this first step the active ODE constraints are grouped by their BMC unwinding depths and common variables. For a BMC depth with active ODE constraints, the ODE constraints can either be grouped into one ODE problem or (in order to reduce dimensionality) partitioned into essentially disjoint ODE problems which only share the same time of interest.

We have embedded the ODE enclosure mechanisms into the constraint solver iSAT [7] as follows: whenever constraint propagation based on equations and bounds cannot perform any further arithmetic deductions, we generate the ODE problems and their inverses for one BMC unwinding depth. Enclosures are then calculated for them, in a round-robin fashion. Thereby, new bounds on the postbox deduced by forward propagation through an ODE problem $\mathcal{P}$ can subsequently be used to also tighten the startbox by propagation through its inverse $\mathcal{P}^{-1}$ and vice versa. When the deduced bounds cease to become tighter, they are returned to interval constraint propagation to allow further arithmetic propagations. During ODE-based interval narrowing, the enclosure mechanism may encounter a conflict when no trajectory connecting the pre- and postbox exists. In that case, a conflict clause is added to the constraint system, such that the solver learns to never again assign this combination of activated triggers and boxes. Deductions from the ODE problems and from the arithmetic constraints thereby propagate new bounds through the entire constraint system. Only if neither arithmetic nor ODE deductions are possible, a decision step is performed.

5 First experimental results

In order to test the presented ideas, we have implemented the method described in the previous section by straightforward integration into iSAT. This integration is prototypical, lacking any optimizations like reuse of inferences along the isomorphic copies of the transition relation in a BMC problem [6]. Given the extremely high computational cost of computing an interval enclosure of an ODE, such mechanisms for copying inferences across isomorphic sub-formulae rather than recomputing them should provide large speedups. Without such optimizations, performance figures like runtime and memory consumption are not indicative of the actual performance of the algorithm. The current implementation can, however, serve as a proof of concept that a tight integration of interval-based ODE-enclosures as yet another interval narrowing operator in interval constraint propagation provides a viable alternative to conventional schemes of hybrid system analysis, where computation of ODE images and transition images are strictly separate phases. When reporting on benchmarks, we will therefore concentrate on (a) the precision of the enclosures generated and (b) the number of hand-overs between equation-based interval constraint propagation (the source of the equations being the transitions and state invariants

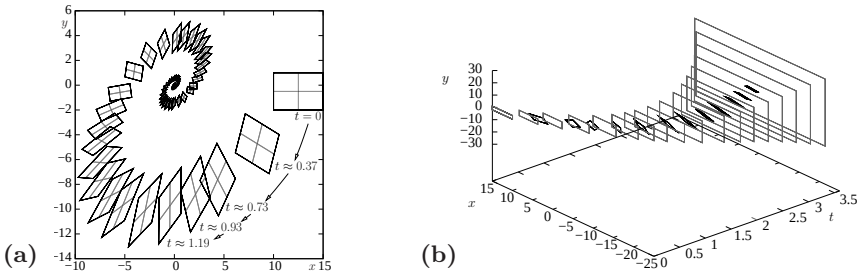

Fig. 3. Damped oscillator. (a) enclosures at intermediate time instants, (b) interval enclosures obtained with and without coordinate transformation (inner/outer sequence)

of the hybrid system) and ODE-enclosure-based interval constraint propagation. Both figures are indicating effectiveness of the proposed strategy.

Damped oscillator. Tightness of the interval enclosures, as expected for at least linear ODEs, can be demonstrated on a damped harmonic oscillator $\frac{dx}{dt} = y - 0.8 \cdot x$ and $\frac{dy}{dt} = -x + 0.3 \cdot y$. While mere interval-based numeric overapproximation used in previous ICP-based approaches to reasoning about ODEs (e.g., [10]) does not accurately reflect the damping, yielding enclosures like the outer one in Fig. 3 (b), our approach yields a reasonably exact overapproximation of the pre-post-relation mediated by the ODE (Fig. 3 (a) and (b), inner sequence).

Bouncing ball. The bouncing ball is a simple, classical example of a hybrid system, suitable as a test for the handover between the different interval narrowing mechanisms. In free fall, height h and speed v of the ball are governed by $\frac{dh}{dt} = v$ and $\frac{ds}{dt} = -g$ where g is the gravitational constant. Whenever the ball hits the ground at $h = 0$, it bounces by discontinuous change of the sign of v. Searching for a ground impact at a time $t \geq 8$ starting from a limited start height, solving required 664 hand-overs between equation-based and ODE-based interval narrowing, entailing the computation of 1754 ODE enclosures which delivered 55 tightened intervals and 5 conflicts[4], the latter being memorized through conflict-driven learning and thus eliminating multiple candidate traces.

6 Conclusion

Within this paper, we have presented a seamless integration of safe numeric integration of ODEs into SAT-modulo-theory (SMT) solving. From the practical point of view, such an integration extends the scope of SMT algorithms from mixed arithmetic-Boolean problems involving relations defined by arithmetic inequations to problems additionally comprising relations defined by initial value problems of ODEs, thus permitting the direct application of SMT to hybrid systems without the need for a preprocessing step replacing ODEs with pre-post-relations defined by (in-)equations. Technically, it involves the embedding

[4] i.e., proofs that the gap between the set of endpoints of one and startpoints of another partial trace cannot be bridged by any continuous trajectory

of interval-based safe numeric approximation of ODE images and ODE pre-images as a further rule for theory propagation in SMT solving.

First experiments show that such an integration is technically feasible and delivers the desired automated deduction system for reasoning about hybrid systems, yet do also indicate that the computational cost of the individual ODE-related deductions is extremely high. The next release will thus drastically reduce their frequency through proven methods for reuse of deductions within isomorphic subformulae [6] in order to attain performance competitive with existing tools optimized for the domain.

References

1. G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani. Verifying industrial hybrid systems with MathSAT. *ENTCS*, 89(4), 2004.
2. F. Benhamou and L. Granvilliers. Continuous and interval constraints. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, Foundations of Artificial Intelligence, chapter 16, pages 571–603. Elsevier, 2006.
3. A. Biere, A. Cimatti, and Y. Zhu. Symbolic model checking without BDDs. In *TACAS'99*, volume 1579 of *LNCS*. Springer-Verlag, 1999.
4. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.
5. B. Dutertre and L. de Moura. A Fast Linear-Arithmetic Solver for DPLL(T). In *Computer-Aided Verification*, volume 4144 of *LNCS*, pages 81–94. Springer, 2006.
6. M. Fränzle and C. Herde. HySAT: An efficient proof engine for bounded model checking of hybrid systems. *Formal Methods in Syst. Design*, 30(3):179–198, 2007.
7. M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *JSAT Special Issue on Constraint Programming and SAT*, 1:209–236, 2007.
8. H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. DPLL(t): Fast decision procedures. In *CAV'04*, volume 3114 of *LNCS*. Springer-Verlag, 2004.
9. T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HYTECH: Hybrid systems analysis using interval numerical methods. In *HSCC'00*, volume 1790 of *LNCS*, pages 130–144. Springer-Verlag, 2000.
10. T. Hickey and D. Wittenberg. Rigorous modeling of hybrid systems using interval arithmetic constraints. In R. Alur and G. J. Pappas, editors, *HSCC'04*, volume 2993 of *LNCS*. Springer, 2004.
11. R. J. Lohner. Enclosing the solutions of ordinary initial and boundary value problems. In *Computerarithmetic: Scientific Computation and Programming Languages*, pages 255–286. Teubner, Stuttgart, 1987.
12. R. E. Moore. Automatic local coordinate transformation to reduce the growth of error bounds in interval computation of solutions of ordinary differential equations. In L. B. Ball, editor, *Error in Digital Computation*, volume II, pages 103–140. Wiley, New York, 1965.
13. R. E. Moore. *Interval Analysis*. Prentice Hall, NJ, 1966.
14. O. Stauning. *Automatic Validation of Numerical Solutions*. PhD thesis, Danmarks Tekniske Universitet, Kgs. Lyngby, Denmark, 1997.