Title

SMC

ISBN

978-1-4503-4590-3

Authors

Shoukry, Yasser

Nuzzo, Pierluigi

Sangiovanni-Vincentelli, Alberto L

et al.

Publication Date

2017-04-13

DOI

10.1145/3049797.3049819

Peer reviewed

SMC: Satisfiability Modulo Convex Optimization

Yasser Shoukry[§†] Pierluigi Nuzzo[*] Alberto L. Sangiovanni-Vincentelli[†]
Sanjit A. Seshia[†] George J. Pappas[**] Paulo Tabuada[§]

[†]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA
[§]Department of Electrical Engineering, University of California, Los Angeles, CA
[*] Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA
[**]Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA

ABSTRACT

We address the problem of determining the satisfiability of a Boolean combination of convex constraints over the real numbers, which is common in the context of hybrid system verification and control. We first show that a special type of logic formulas, termed monotone Satisfiability Modulo Convex (SMC) formulas, is the most general class of formulas over Boolean and nonlinear real predicates that reduce to convex programs for any satisfying assignment of the Boolean variables. For this class of formulas, we develop a new satisfiability modulo convex optimization procedure that uses a lazy combination of SAT solving and convex programming to provide a satisfying assignment or determine that the formula is unsatisfiable. Our approach can then leverage the efficiency and the formal guarantees of state-of-the-art algorithms in both the Boolean and convex analysis domains. A key step in lazy satisfiability solving is the generation of succinct infeasibility proofs that can support conflict-driven learning and decrease the number of iterations between the SAT and the theory solver. For this purpose, we propose a suite of algorithms that can trade complexity with the minimality of the generated infeasibility certificates. Remarkably, we show that a minimal infeasibility certificate can be generated by simply solving one convex program for a sub-class of SMC formulas, namely ordered positive unate SMC formulas, that have additional monotonicity properties. Perhaps surprisingly, ordered positive unate formulas appear themselves very frequently in a variety of practical applications. By exploiting the properties of monotone SMC formulas, we can then build and demonstrate effective and scalable decision procedures for problems in hybrid system verification and control, including secure state estimation and robotic motion planning.

1. INTRODUCTION

The central difficulty in analyzing and designing hybrid systems is the very different nature of the technical tools used to analyze continuous dynamics (e.g., real analysis) and discrete dynamics (e.g., combinatorics). The same difficulty arises in the context of optimization and feasibility problems involving continuous and discrete variables. In fact, some of these problems arise from the analysis of hybrid systems. In complex, high-dimensional systems, a vast discrete/continuous space must be searched under constraints that are often nonlinear. Developing efficient techniques to perform this task is, therefore, crucial to substantially enhance our ability to design and analyze hybrid systems.

Constraint Programming (CP) and Mixed Integer Programming (MIP) have emerged over the years as means for addressing many of the challenges posed by hybrid systems and hybrid optimization problems [1]. Rooted in Satisfiability (SAT) solving and, more recently, Satisfiability Modulo Theory (SMT) solving, CP relies on logic-based methods such as domain reduction and constraint propagation to accelerate the search. Modern SAT and SMT solvers [2] can efficiently find satisfying valuations of very large propositional formulas with complex Boolean structure, including combinations of atoms from various decidable theories, such as lists, arrays, bit vectors, linear integer arithmetic, and linear real arithmetic. However, while SMT solving for generic nonlinear theories over the reals is undecidable in general [3, 4], algorithms and tools that can address useful fragments of these theories with solid guarantees of correctness and scalability have only recently started to appear.

MIP-based approaches encode, instead, a Boolean combination of nonlinear constraints into a conjunction of mixed integer constraints and solve it by leveraging numerical algorithms based on branch-and-bound and cutting-plane methods. When applied to mixed integer convex constraints, optimization-based techniques tend to be efficient if the Boolean structure of the problem is simple. Moreover, convex programming is extensively used as the core engine in a variety of applications, ranging from control design to communications, from electronic design to data analysis and modeling [5]. However, encoding some logic operations, such as disjunction and implication, into mixed integer constraints usually requires approximations and heuristic techniques, such as the well-known "big-M" method [1], which may eventually affect the correctness of the solution.

In this paper, we aim at bridging the gap between CP and MIP based techniques, which have shown superior performance in handling, respectively, complex Boolean structures and large sets of convex constraints. While attempts at combining logic-based inference with optimization trace back to the 1950s [1], they were mostly limited to "prepro-

HSCC'17, April 18 - 20, 2017, Pittsburgh, PA, USA

ISBN 978-1-4503-4590-3/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3049797.3049819

cessing" and "implicit enumeration" schemes, doomed soon to be outperformed by branch-and-bound or cutting-plane methods. As the effectiveness of CP techniques has steadily increased over the years, their integration with optimization has been the subject of increasing research activity. However, devising a robust and widely acceptable integration scheme is still largely an open issue. We tackle this challenge by focusing on the satisfiability problem for a class of formulas over Boolean variables and convex constraints. We target this special class of problems because of their pervasiveness as well as the efficiency and robustness of the solution methods made available by convex programming [5].

The first contribution of this paper is to identify a special type of logic formulas, termed monotone Satisfiability Modulo Convex (SMC) formulas, and to show that it is the most general class of formulas over Boolean and nonlinear real arithmetic predicates that can be solved via a finite number of convex programs. For monotone SMC formulas, we develop a new procedure, which we call Satisfiability Modulo Convex Optimization, that uses a lazy combination of SAT solving and convex programming to provide a satisfying assignment or determine that the formula is unsatisfiable. As in the lazy SMT paradigm [2], a classic SAT solving algorithm [6] interacts with a theory solver. The SAT solver efficiently reasons about combinations of Boolean constraints to suggest possible assignments. The theory solver only checks the consistency of the given assignments, i.e., conjunctions of theory predicates, and provides the reason for the conflict, an UNSAT certificate, whenever inconsistencies are found.

Checking the feasibility of a set of convex constraints can be performed efficiently, with a complexity that is polynomial in the number of constraints and real variables. A key step is, however, the generation of compact certificates to support conflict-driven learning and decrease the number of iterations between the SAT and the theory solver. The second contribution of this paper is to propose a suite of algorithms that can trade complexity with the minimality of the generated certificates. Remarkably, we show that a minimal infeasibility certificate can be generated by simply solving one convex program for a sub-class of monotone SMC formulas, namely Ordered Positive Unate (OPU) formulas, that present additional monotonicity properties. Since monotone SMC and OPU formulas appear frequently in practical applications, we can then build and demonstrate effective and scalable decision procedures for several problems in hybrid system verification and control. Experimental results show that our approach outperforms state-of-the-art SMT and MICP solvers on problems with complex Boolean structure and a large number of real variables.

Related Work. Our work focuses on feasibility problems and leverages optimization methods to accelerate the search task. In this respect, it differs from other research efforts such as the "optimization modulo theories" [7] or "symbolic optimization" [8] approaches, which propose SMT-based techniques to solve optimization problems. The ABSOLVER tool [9] adopts a similar lazy SMT approach as in our work, by leveraging a generic nonlinear optimization tool to solve Boolean combinations of polynomial arithmetic constraints. However, generic nonlinear optimization techniques may produce incomplete or possibly incorrect results, due to their "local" nature, explicitly requiring upper and lower bounds to all the real variables. The Z3 [10] solver can also provide support for nonlinear polynomial arithmetic, while still being subject to incompleteness or termination issues[1]. The iSAT algorithm builds on a unification of SAT-solving and Interval Constraint Propagation (ICP) [11] to efficiently address arbitrary smooth, possibly transcendental, functions. The integration of SAT solving with ICP is also used in dReal to build a δ-complete decision procedure which solves SMT problems over the reals with nonlinear functions, such as polynomials, sine, exponentiation, or logarithms [12], but with limited support for Boolean combinations of nonlinear constraints. By targeting the special classes of convex constraints and monotone SMC formulas, we are able to leverage the efficiency, robustness, and correctness guarantees of state-of-the-art convex optimization algorithms. Moreover, we can efficiently generate UNSAT certificates that are more compact, or even minimal.

Our results build upon the seminal work of CalCS, which pioneered the integration of SAT solving and optimization algorithms for convex SMT formulas [13]. Differently from CalCS, we focus on the satisfiability problem for monotone SMC formulas, which do not require approximation techniques to handle negated convex constraints and are rich enough to capture several problem instances in hybrid system verification and control. For SMC formulas, we provide formal correctness guarantees for our algorithms in terms of δ-completeness [4]. Moreover, we propose new algorithms to generate UNSAT certificates that improve on the efficiency or minimality guarantees of the previous ones, which were based on the sensitivity of the objective of a convex optimization problem to its constraints.

We have also recently developed specialized SMT-based algorithms for applications in secure state estimation, Imhotep-SMT [14], and robotic motion planning [15]. We show that the approach detailed in this paper subsumes these results. Finally, our decision procedure encompasses Mixed Integer Convex Programming (MICP) based techniques. In fact, we show that any feasibility problem on MIC constraints can be posed as a satisfiability problem on a monotone SMC formula. While an MICP formulation can execute faster on problems with simpler Boolean structure, our algorithms outperform MICP-based techniques on problems with large numbers of Boolean variables and constraints.

2. MOTIVATING EXAMPLE

We illustrate the practical relevance of the logic addressed in this paper using a representative hybrid system control problem inspired by robotic motion planning [15]. To develop algorithmic techniques for robotic motion planning, we need to reason about the tight integration of discrete abstractions (as in *task planning*) with continuous motions (*motion planning*) [16]. Task planning relies on specifications of temporal goals that are usually captured by logics such as Linear Temporal Logic (LTL) [17]. Motion planning deals with complex geometries, motion dynamics, and collision avoidance constraints that can only be accurately captured by continuous models. Ideally, we wish to combine effective discrete planning techniques with effective methods for generating collision-free and dynamically-feasible trajectories to satisfy both the dynamics and task planner constraints.

[1] As reported by the official Z3 website, http://research.microsoft.com/en-us/um/redmond/projects/z3/arith-tutorial/.

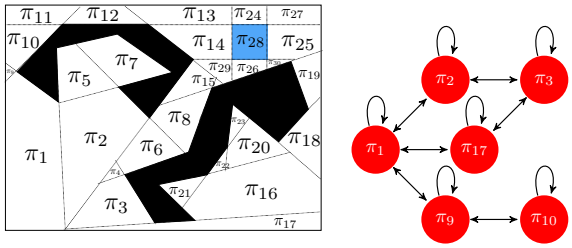

Figure 1: Obstacle-based discretization of the workspace for the motion planning problem (left), and transition system describing the adjacency relation between regions in the workspace (right).

For simplicity, we present below an encoding for the basic reach-avoid problem. We assume a discrete-time, linear model of the robot dynamics and a description of a workspace in terms of a set of obstacles and a target region, where both the obstacles and the region are polyhedra. The goal is to construct a trajectory, and the associated control strategy, that steers the robot from its initial point to the target while avoiding obstacles. We further assume that the workspace is partitioned into a set of regions, as in Figure 1 (left), which can also be described by polyhedra and captured by linear constraints of the form $(Px + q \leq 0)$, where $x \in \mathbb{R}^n$ represents the state variables of the robot, including its coordinates in the workspace. For a fixed horizon L, the controller design problem translates into *finding a sequence of regions of length L (discrete plan) that brings the robot from the initial point to the target and is compatible with the continuous dynamics.*

It is convenient to capture the adjacency relation between regions via a transition system as in Figure 1 (right). A valid trajectory for the robot can then be represented by a run of the transition system. Let b_i^j be a Boolean variable that is asserted if and only if the robot is in region i at time j. We can then encode the constraints for the controller using the following *logic formula* φ:

$$
\begin{aligned}
&\varphi ::= b^0_{\text{start}} && \text{(Initial partition)}\\
&\wedge\, b^L_{\text{goal}} && \text{(Goal partition)}\\
&\wedge \left(b_i^j \rightarrow \bigvee_{i' \in \Pi(i)} b_{i'}^{j+1} \right) && \forall\, j \in \{0,\dots,L-1\}, i \in \{1,\dots,m\}\\
& && \text{(Adjacency constraints)}\\
&\wedge \left(\sum_{i=1}^{m} b_i^j = 1 \right) && \forall j \in \{0,\dots,L-1\}\\
& && \text{(Mutual exclusion)}\\
&\wedge \left(x^{j+1} = Ax^j + Bu^j \right) && \forall j \in \{0,\dots,L-1\}\\
& && \text{(Robot dynamics)}\\
&\wedge \left(\|u^j\| \leq \overline{u} \right) && \forall j \in \{0,\dots,L-1\}\\
& && \text{(Input constraints)}\\
&\wedge \left(x^0 = \overline{x} \right) && \text{(Initial state)}\\
&\wedge \left(b_i^j \rightarrow P_i x + q_i \leq 0 \right) && \forall j \in \{0,\dots,L-1\}, i \in \{1,\dots,m\}\\
& && \text{(Region constraints)}
\end{aligned}
$$

where $\Pi(i)$ is the set of regions that are adjacent to region i, m is the total number of regions, A and B are the state and input matrices governing the robot dynamics, and $\overline{u}$ is the maximum feasible magnitude $\|u^j\|$ (e.g., ℓ_2 or ℓ_∞ norm) of the control input at time j.

The formula φ captures the constraints of a reach-avoid problem as a conjunction of logic clauses, possibly including pseudo-Boolean predicates (e.g., mutual exclusion or cardinality constraints), and where some of the literals are convex constraints (e.g., in the dynamics, input, and region constraints). We call such a formula a *monotone SMC formula* since none of the convex constraints are negated. The formal definition is in Section 3. We further observe that the satisfying assignments of the "purely Boolean" portion of φ are characterized by an ordering imposed by the feasible runs of the transition system in Figure 1. If a sequence of regions σ is feasible, then so is any prefix sequence of σ. We will call the formulas encoding such a scenario *OPU formulas* and provide the formal definition in Section 5.3. OPU formulas are by no means specific to the encoding of motion planning constraints. They appear in several applications; for example, whenever Boolean variables are used to capture the occurrence of events (or modes) that are sequentially concatenated. This is the case for the variables encoding the states in a finite state machine or for switched systems in which modes are captured by a finite state automaton and dynamics are expressed by convex constraints.

We will show that *Boolean solving can be effectively combined with convex optimization to determine the satisfiability of monotone SMC and OPU formulas. Scalable decision procedures can be developed based on efficient methods for detecting minimal sets of conflicting convex constraints. In particular, this task reduces to solving only one convex program in OPU formulas.* To formalize these categories of decision problems, we first define the syntax and semantics of monotone SMC formulas.

3. MONOTONE SMC FORMULAS

3.1 Notation

We denote as $b = (b_1, b_2, \dots, b_m)$ the set of Boolean variables in a formula, with $b_i \in \mathbb{B}$, and with $x = (x_1, x_2, \dots, x_n)$ the set of real-valued variables, where $x_i \in \mathbb{R}$. When not directly inferred from the context, we adopt the notation $\varphi(x, b)$ to highlight the set of variables over which a formula φ is defined. A valuation μ is a function that associates each variable in b to a truth value in $\mathbb{B}$. We denote as $[\![b]\!]_\mu \in \mathbb{B}^m$ the set of values assigned to each variable in b by μ.

A *convex constraint* is a constraint of the form $f(x)\ \{<, \leq\}\ 0$ or $h(x) = 0$, where $f(x)$ and $h(x)$ are convex and affine (linear) functions, respectively, of their real variables $x \in \mathcal{D} \subseteq \mathbb{R}^n$, $\mathcal{D}$ being a convex set. In what follows, we will compactly denote a generic convex constraint as $(g(x) \lhd 0)$. A convex constraint is associated with a set $\mathcal{C} = \{x \in \mathcal{D} : g(x) \lhd 0\}$, i.e., the set of points in the domain of the convex function g that satisfy the constraint. The set $\mathcal{C}$ is also convex[2]. We further denote the negation of a convex constraint, expressed in the form $f(x) \geq 0$ $(f(x) > 0)$, as *reverse convex* constraint. A reverse convex constraint is, in general, non-convex and so is its satisfying set. For a formal definition of convex function, we refer the reader to the literature [5].

[2]In fact, given a representation of the convex domain $\mathcal{D}$ as a convex constraint $(d(x) \leq 0)$, we can directly account for the domain by directly embedding it into the expression of the convex constraint, e.g., by defining $(\tilde{g}(x) \lhd 0) = (g(x) \lhd 0) \wedge (d(x) \leq 0)$.

To be able to capture linear constraints on Boolean variables in a compact way, we also use pseudo-Boolean predicates. A *pseudo-Boolean predicate pB_predicate* is an affine constraint over Boolean variables with integer coefficients.

3.2 Syntax and Semantics

We represent *monotone Satisfiability Modulo Convex (SMC) formulas* to be quantifier-free formulas in conjunctive normal form, with atomic propositions ranging over propositional variables and convex constraints. Formally,

Definition 3.1 (Monotone SMC Formula). *A monotone SMC formula is any formula that can be represented using the following syntax:*

$$\begin{aligned} formula &::= \{clause \wedge\}^* clause \\ clause &::= (\{literal \vee\}^* literal) \mid pB\_predicate \\ literal &::= bool\_var \mid \neg bool\_var \mid \top \mid \bot \mid \\ &\quad conv\_constraint \\ conv\_constraint &::= equation \mid inequality \\ equation &::= affine\_function = 0 \\ inequality &::= convex\_function\ relation\ 0 \\ relation &::= < \mid \leq \end{aligned} \tag{1}$$

In the grammar above, *bool_var* denotes a Boolean variable, and *affine_function* and *convex_function* denote affine and convex functions, respectively. Monotone SMC formulas can only admit convex constraints as theory atoms. Differently from generic (non-monotone) SMC formulas, i.e., generic SMT formulas over convex constraints [13], reverse convex constraints are not allowed. The monotonicity property is key to guarantee that a *model*, i.e., a satisfying assignment, can always be found by solving one (or more) optimization problems that are convex, as we further discuss below. We rely on the disciplined convex programming approach [5, 18] as an effective method to specify the syntax of convex constraints out of a library of atomic functions and automatically ensure the convexity of a constraint.

Formulas are interpreted over valuations μ (i.e., $[\![b, x]\!]_\mu \in \mathbb{B}^m \times \mathbb{R}^n$). A formula φ is satisfied by a valuation μ ($\mu \models \varphi$) if and only if all its clauses are satisfied, that is, if and only if at least one literal is satisfied in any clause. A Boolean literal l is satisfied if $[\![l]\!]_\mu = \top$. Satisfaction of real constraints is with respect to the standard interpretation of the arithmetic operators and the ordering relations over the reals.

Aiming at a scalable solver architecture, we exploit efficient numerical algorithms based on convex programming to decide the satisfiability of convex constraints and provide a model when the constraints are feasible. However, convex solvers usually perform floating point (hence inexact) calculations, although the numerical error can be bounded by a constant that can be made arbitrarily small. Therefore, to provide correctness guarantees for our algorithms, we resort to similar notions of δ-satisfaction and δ-completeness as the ones previously proposed by Gao et al. [4], which we define below on generic SMC formulas.

Definition 3.2 (δ-Relaxation). *Given an SMC formula φ, let $|C|$ be the number of convex constraints in φ and $\delta \in \mathbb{Q}^+ \cup \{0\}$ any non-negative rational number δ. We define the δ-relaxation of φ as the formula obtained by replacing any convex constraints of the forms $(f_i(x) \leq 0)$ and $(h_j(x) = 0)$ in φ with their perturbed versions $(f_i(x) \leq \delta_i)$ and $(|h_j(x)| \leq \delta_j)$, respectively, where $\delta_k \in \mathbb{Q}^+ \cup \{0\}$ for all $k \in \{1, \ldots, |C|\}$, and such that $\sum_{k=1}^{|C|} \delta_k \leq \delta$. We denote the newly obtained formula as φ^δ.*

Definition 3.3 (δ-Satisfaction). *Given an SMC formula φ and $\delta \in \mathbb{Q}^+$, we say that φ is δ-`SAT` when φ^δ is satisfiable. Otherwise, we say that φ is `UNSAT`.*

We simply say that φ is `SAT` when there is no ambiguity about the choice of δ. If $\varphi = \varphi^0$ is satisfiable, then φ^δ is satisfiable for all $\delta \in \mathbb{Q}^+$, i.e., $\varphi \rightarrow \varphi^\delta$. The opposite is, however, not true. In fact, depending on the value of δ, φ^0 and φ^δ can be made, respectively, false and true at the same time. When this happens, we admit both the `SAT` and `UNSAT` answers. This outcome is acceptable in practical applications, since small perturbations capable of modifying the truth value of a formula usually denote lack of robustness either in the system or in the model. Finally, we say that an algorithm is *δ-complete* if it can correctly solve the satisfiability problem for an SMC formula in the sense of Definition 3.3.

3.3 Properties

Monotone SMC formulas have the desirable property that they can always be solved via a finite set of convex feasibility problems. To show this, we introduce the following proposition and the related definitions of *Boolean abstraction* and *monotone convex expansion* of a convex formula.

Definition 3.4 (Monotone Convex Expansion). *Let φ be an SMC formula, C be the set of convex constraints, and $|C|$ its cardinality. We define* propositional abstraction *of φ a formula φ_B obtained from φ by replacing each convex constraint with a Boolean variable a_i, $i \in \{1, \ldots, |C|\}$. We further define* monotone convex expansion *of φ the formula φ' defined as:*

$$\varphi' = \varphi_B \wedge \bigwedge_{i=1}^{|C|} (a_i \rightarrow (g_i(x) \lhd 0)), \tag{2}$$

where $(g_i(x) \lhd 0)$ denotes a convex constraint as defined in Section 3.1.

Proposition 3.5. *Let φ' be the monotone convex expansion of a monotone SMC formula φ, defined as in* (2)*, where φ_B is the propositional abstraction of φ. Then, the following properties hold:*

1. *φ and φ' are equisatisfiable, i.e., if (b^*, x^*, a^*) is a model (a satisfying assignment) for φ', then (b^*, x^*) is a model for φ; if φ' is unsatisfiable, then so is φ;*
2. *any satisfying Boolean assignment for φ_B turns φ' into a conjunction of convex constraints;*
3. *the satisfiability problem for φ', hence φ, can always be cast as the feasibility problem for a finite disjunction of convex programs.*

Proposition 3.5 directly descends from the monotonicity of φ. Its proof as well as the proofs of all the results in this paper can be found in an extended version of the paper [19]. By Proposition 3.5, any monotone SMC formula φ can be solved by casting and solving a disjunction of convex programs. We will use this property to construct our decision procedure in Section 4. It is possible to show that monotone convex formulas are also the only class of formulas over

Boolean propositions, pseudo-Boolean predicates, and predicates in the nonlinear theories over the reals, to present this property. This is formally stated by the following theorem.

Theorem 3.6. *Let φ be a formula over Boolean propositions, pseudo-Boolean predicates, and predicates in the nonlinear theories over the reals, and such that the satisfiability problem can be posed as the feasibility problem for a finite disjunction of convex programs. Then, φ can be posed as a monotone SMC formula.*

Finally, the following corollary is an immediate consequence of the results above.

COROLLARY 3.7. *Monotone SMC formulas include any Boolean Satisfiability (SAT) problem instance and any Mixed Integer Convex (MIC) feasibility problem instance as a particular case.*

Any MIC formulation can be translated into an equisatisfiable SMC formula, but the opposite is not true. Often, disjunctions of predicates, such as the one in $\varphi := \neg b \vee (x-3<0)$, cannot be expressed as a conjunction of MIC constraints unless relaxations (approximations) are used [1]. For instance, φ is typically encoded with the constraint $c := x-3 < (1-b)\cdot M$, using the "big-M" method. However, for any value of M, the assignment $(b,x)=(0,M+3)$ is a satisfying assignment for φ, but violates c.

4. ALGORITHM ARCHITECTURE

Our decision procedure combines a SAT solver (SAT-SOLVE) and a theory solver ($\mathcal{C}$-SOLVE) for convex constraints on real numbers by following the *lazy* SMT paradigm [2]. The SAT solver efficiently reasons about combinations of Boolean and pseudo-Boolean constraints, using the David-Putnam-Logemann-Loveland (DPLL) algorithm [6], to suggest possible assignments for the convex constraints. The theory solver checks the consistency of the given assignments and provides the reason for a conflict, i.e., an `UNSAT` *certificate*, whenever inconsistencies are found. Each certificate results in learning new constraints which will be used by the SAT solver to prune the search space. Because the monotone convex expansion φ' of a monotone formula φ translates into a conjunction of convex constraints for any Boolean assignments by Proposition 3.5, we are assured that we can solve for φ using a lazy SMT approach, since we are guaranteed to generate queries to the theory solver that are always in the form of conjunctions of convex constraints and, therefore, can be efficiently solved by convex programming. Our decision task is thus broken into two simpler tasks, respectively, over the Boolean and convex domains.

As illustrated in Algorithm 4, we start by generating the propositional abstraction $\varphi_B(b,a)$ of φ. We denote as $\mathcal{M}$ the map that associates each convex constraint in φ to an auxiliary variable a_i. By only relying on the Boolean structure of φ_B, SAT-SOLVE may either return `UNSAT` or propose a satisfying assignment μ for the variables b and a, thus hypothesizing which convex constraints should be jointly satisfied.

Let a^* be the assignment proposed by SAT-SOLVE for the auxiliary Boolean variables a in φ_B; we denote as $\mathrm{supp}(a^*)$ the set of indices of auxiliary variables a_i which are asserted in a^*, i.e., such that $[\![a_i]\!]_\mu = \top$. This Boolean assignment is then used by $\mathcal{C}$-SOLVE to determine whether there exist

Algorithm 1 SMC
Input: φ, δ **Output:** $\eta(b,x)$

```
1: (φ_B(b,a), M) := ABSTRACT(φ);
2: while TRUE do
3:    (status, μ(b,a)) := SAT-SOLVE(φ_B);
4:    if status == UNSAT then
5:       return
6:    else
7:       (status, x) := C-SOLVE.CHECK(μ, M, δ);
8:       if status == SAT then
9:          return η(b,x)
10:      else
11:         φ_ce := C-SOLVE.CERT(μ, M, δ);
12:         φ_B := φ_B ∧ φ_ce;
```

real variables $x \in \mathbb{R}^n$ which satisfy all the convex constraints related to asserted auxiliary variables. Formally, we are interested in the following problem

$$\text{find} \quad x \quad \text{s.t.} \quad g_i(x) \lhd 0 \qquad \forall\, i \in \mathrm{supp}(a^*) \tag{3}$$

which we denote as feasibility problem associated with a^*. The above problem can be efficiently cast as the following optimization problem with the addition of slack variables, which we call a *sum-of-slacks feasibility* (*SSF*) problem:

$$\min_{\substack{s_1,\ldots,s_L \in \mathbb{R} \\ x \in \mathbb{R}^n}} \sum_{i=1}^{L} |s_i| \quad \text{s.t.} \qquad g_{j_i}(x) \lhd s_i, \quad i = 1,\ldots,L \tag{4}$$

where L is the cardinality of $\mathrm{supp}(a^*)$ and j_i spans $\mathrm{supp}(a^*)$ as i varies in $\{1,\ldots,L\}$. Problem 4 is equivalent to (3), as it tries to minimize the infeasibilities of the constraints by pushing each slack variable to be as much as possible close to zero. The optimum is zero and is achieved if and only if the original set of constraints in (3) is feasible. Therefore, if the cost at optimum is zero (in practice, the condition $\sum_{i=1}^{L} |s_i| \leq \delta$ is satisfied for a "small" $\delta \in \mathbb{Q}^+$), then μ is indeed a valid assignment, an optimum x^* is found, and our algorithm terminates with `SAT` and provides the solution (x^*, b). Otherwise, an UNSAT certificate φ_{ce} is generated in terms of a new Boolean clause explaining which auxiliary variables should be negated since the associated convex constraints are conflicting. A trivial certificate can always be provided in the form of:

$$\varphi_{\text{trivial-ce}} = \bigvee_{i \in \mathrm{supp}(a^*)} \neg a_i, \tag{5}$$

which encodes the fact that at least one of the auxiliary variables indexed by an element in $\mathrm{supp}(a^*)$ should actually be negated. The augmented Boolean problem consisting of the original formula φ_B and the generated certificate φ_{ce} is then fed back to SAT-SOLVE to produce a new assignment. The sequence of new SAT queries is then repeated until either $\mathcal{C}$-SOLVE terminates with `SAT` or SAT-SOLVE terminates with `UNSAT`. The following statement summarizes the formal guarantees of Algorithm 4 with the trivial certificate (5).

PROPOSITION 4.1. *Let φ be a monotone SMC formula and $\delta \in \mathbb{Q}^+$ a user-defined tolerance used in $\mathcal{C}$-SOLVE.CHECK in Algorithm 4 to accommodate numerical errors. Algorithm 4 with the UNSAT certificate φ_{ce} in* (5) *is δ-complete.*

The worst case bound on the number of iterations in Algorithm 4 is exponential in the number of convex constraints

Certificate	# Convex Programs	Length ℓ
Trivial	1	$\lvert S\rvert$
IIS-Based	Exponential in $\lvert S\rvert$	$\lvert I\rvert$ (minimal)
SSF-Based	Linear in $\lvert S\rvert$	$\lvert I\rvert \leq \ell \leq \lvert S\rvert$
Prefix-Based	1	$\lvert I\rvert \leq \ell \leq \lvert S\rvert$

Table 1: Proposed algorithms for certificate generation: number of convex programs needed to generate the certificate and length of the generated certificate. |S| is the number of constraints in the convex program.

$|C|$. To help the SAT solver quickly find a correct assignment, a central problem in the lazy SMT paradigm is to generate *succinct certificates*, possibly highlighting the minimum set of conflicting assignments, i.e., the "reason" for the inconsistency. The smaller the *conflict clause*, the larger is the region that is excluded from the search space of the SAT solver. Moreover, certificates should be generated *efficiently*, ideally in polynomial time, to provide a negligible overhead with respect to the exponential complexity of SAT solving. In the following, we discuss efficient algorithms to generate smaller conflict clauses.

5. GENERATING SMALL CERTIFICATES

5.1 IIS-Based Certificates

When $\mathcal{C}$-Solve.Check finds an infeasible problem, a minimal certificate can be generated by providing an *Irreducibly Inconsistent Set* (IIS) [20] of constraints, defined as follows.

Definition 5.1 (Irreducibly Inconsistent Set). *Given a feasibility problem with constraint set S, an* Irreducibly Inconsistent Set *I is a subset of constraints $I \subseteq S$ such that: (i) the feasibility problem with constraint set I is infeasible; (ii) $\forall\ c \in I$, the feasibility problem with constraint set $I \setminus \{c\}$ is feasible.*

In other words, an IIS is an infeasible subset of constraints that becomes feasible if any single constraint is removed. Let $\mathcal{I}$ be set of indices of auxiliary Boolean variables in φ_B that are associated to a convex constraint in an IIS I. Then, once I is found, a minimal certificate can be generated as

$$\varphi_{\text{IIS-ce}} = \bigvee_{i \in \mathcal{I}} \neg a_i. \tag{6}$$

Most of the techniques proposed in the literature to isolate IISs are based on either adding constraints, one by one or in groups, or by deleting them from the original problem [20]. An IIS guarantees that the length of the certificate is minimal, which can dramatically reduce the search space in Algorithm 4. However, isolating one IIS is expensive, especially for nonlinear programs. In the worst case, as shown in Table 1, finding an IIS can require solving a feasibility problem for each subset of constraints in S, which is exponential in the size $|S|$. The following proposition summarizes the correctness guarantees of Algorithm 4 with the IIS-based certificate (6).

Proposition 5.2. *Let φ be a monotone SMC formula and $\delta \in \mathbb{Q}^+$ a user-defined tolerance used in $\mathcal{C}$-Solve.Check in Algorithm 4 to accommodate numerical errors. Algorithm 4 with the UNSAT certificate in* (6) *is δ-complete.*

In the following, we describe an algorithm that can generate a small, albeit non minimal, set of conflicting constraints by solving a number of convex programs that is linear in $|S|$.

Algorithm 2 $\mathcal{C}$-Solve.Cert-SSF$(\mu, \mathcal{M}, \delta)$

Compute optimal slack variables and sort them
  $s^* :=$ Solve-SSF$(\mu, \mathcal{M}, \delta)$;
  $s' :=$ sortAscendingly(s^*);
Pick index for minimum slack
  $\mathcal{I}\_min :=$ Index(s'_1);
  $\mathcal{I}\_max :=$ Index$(s'_{\{|s|,|s|-1,\ldots,2\}})$;
Search linearly for the UNSAT certificate
  status = `SAT`; counter = 1;
  $\mathcal{I}\_temp := \mathcal{I}\_min \cup \mathcal{I}\_max_{counter}$;
while status == `SAT` **do**
    (status, x) := $\mathcal{C}$-Solve.Check$(\mu_{\mathcal{I}_temp}, \mathcal{M}, \delta)$;
    if status == `UNSAT` **then**
      $\varphi_{\text{ce-SSF}} := \bigvee_{i \in \mathcal{I}\_temp} \neg a_i$;
    else
      counter := counter + 1;
      $\mathcal{I}\_temp := \mathcal{I}\_temp \cup \mathcal{I}\_max\_r_{counter}$;
return $\varphi_{\text{ce-SSF}}$

5.2 SSF-Based Certificates

A computationally more efficient alternative to IIS-based certificates is to directly exploit the information in the slacks of the SSF problem (4). If a constraint k is associated with a non-zero optimal slack, $|s^*_k| > 0$, then it is a member of one of the IIS in problem (3). However, the set of all the constraints with a non-zero slack does not necessarily include all the constraints of at least one IIS. Therefore, we propose a search procedure over the constraint set S, which guarantees that at least one IIS is included in the returned set of conflicting constraints by solving a number of convex programs that is linear in the number of constraints $|S|$ in (3), even if the returned conflict set may not be minimal. In fact, the cardinality of the returned conflict set, hence the length of the proposed certificate, can be as large as $|S|$ in the worst case, as shown in Table 1.

The conjecture behind the search strategy is that the constraints with the highest slack values are most likely to be in at least one IIS and conflict with the constraint with the lowest (possible zero) slack. We can then generate a small conflict set including the lowest slack constraint in conjunction with the highest slack constraints, added one-by-one, until a conflict is detected. At each step we solve a convex feasibility problem to detect the occurrence of a conflict. The earlier a conflict is detected, the earlier our search terminates and the shorter the certificate will be. Based on this intuition, our procedure is summarized in Algorithm 2.

We first compute the optimal slacks s^* and sort them in ascending order. We then pick the constraint corresponding to the minimum slack, indexed by $\mathcal{I}\_min$, and generate a new set of indexes $\mathcal{I}\_temp$ by searching for one more constraint that leads to a conflict with the minimum slack constraint, starting with the constraint related to the maximum slack. $\mathcal{I}\_max$ is the set of all slack indexes except the index of the minimum slack in $\mathcal{I}\_min$. If the constraints indexed by $\mathcal{I}\_temp$ are infeasible, then we obtain a conflict set of two elements, and can immediately generate the UNSAT certificate. Otherwise, we repeat the same process by adding the constraint associated with the second largest slack variable in the sorted list of slacks, till we reach a conflicting set. Once the set is discovered, we stop and generate the compact certificate using the auxiliary variables indexed by $\mathcal{I}\_temp$. The following proposition summarizes the correctness guarantees of Algorithm 4 with the SSF-based certifi-

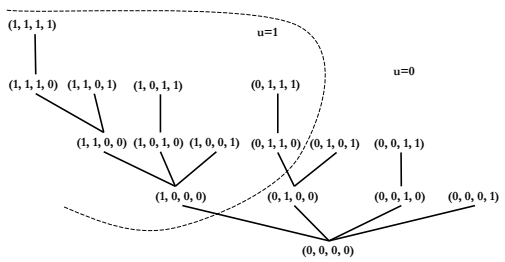

Figure 2: Pictorial representation of the partial order over the assignments for an OPU function u.

cate in Algorithm 2.

PROPOSITION 5.3. *Let φ be a monotone SMC formula and $\delta \in \mathbb{Q}^+$ a user-defined tolerance used in $\mathcal{C}$-SOLVE.CHECK in Algorithm 4 to accommodate numerical errors. Algorithm 4 with the certificate from Algorithm 2 is δ-complete.*

5.3 Prefix-Based Certificate

Under additional monotonicity assumptions on the structure of φ we are able to construct UNSAT certificates that are "minimal" by solving only one convex program. To formalize these monotonicity assumptions and the related notion of minimality, we introduce the concept of *Ordered Positive Unate (OPU) function* as an extension of the classical definition of positive unite function below. For convenience, we also use the notation $b = 1(0)$ to indicate that b is asserted (negated).

DEFINITION 5.4 (POSITIVE UNATE FUNCTION). *A Boolean function $u(b_1, b_2, \ldots, b_m)$ is said to be* positive unate *in b_i if, for all possible values of b_j, $j \neq i$, we have*

$$u(b_1, \ldots, b_{i-1}, 0, b_{i+1}, \ldots, b_m) \leq u(b_1, \ldots, b_{i-1}, 1, b_{i+1}, \ldots, b_m),$$

i.e., by only switching b_i from 0 to 1 we do not decrease the value of u.

DEFINITION 5.5 (OPU FUNCTION). *A Boolean function $u(b_1, b_2, \ldots, b_m)$ is said to be* ordered positive unate *with respect to an ordering $(b_1, \ldots, b_m)$ of its Boolean variables if, for all possible values of b_i, we have*

$$u(0, \ldots, 0) \leq u(b_1, 0, \ldots, 0) \leq u(b_1, b_2, 0, \ldots, 0) \leq \ldots$$
$$\cdots \leq u(b_1, b_2, \ldots, b_{m-1}, 0) \leq u(b_1, b_2, \ldots, b_{m-1}, b_m).$$

In other words, given a valuation over the ordered set $b = (b_1, \ldots, b_m)$, and such that there is a suffix of variables assigned to zero, i.e., $b_i = b_{i+1} = \ldots = b_{m-1} = b_m = 0$, $i \in \{1, \ldots, m\}$, switching the first variable b_i in the suffix from 0 to 1 does not decrease the value of u.

By Definition 5.5, OPU functions determine a partial order over the set of Boolean assignments based on their prefixes, and such that $(0, \ldots, 0)$ is the bottom element. A pictorial representation of such an order is offered in Figure 2 when $m = 4$. All the assignments for $(b_1, \ldots, b_4)$ form a tree; each vertex in the tree shares a prefix with its ancestor and differs from it only in the first variable of the suffix, which is set to 1. We also show in Figure 2 a possible scenario for the values of u. If u evaluates to 1 at a certain vertex v in the tree, then the value of u over all the upper vertices along the path from v to a leaf is also bounded to be 1. Similarly, if u evaluates to 0 at v, then this will be the case over all the lower vertices along the path from v to the root. We now extend this notion of monotonicity to a sub-class of convex formulas of our interest, which we denote as *ordered positive unate formulas*.

DEFINITION 5.6 (OPU FORMULAS). *Let φ be a monotone SMC formula, φ_B its propositional abstraction, and $\chi_\varphi(a_1, a_2, \ldots, a_{|C|})$ be the restriction of the characteristic function for the valid assignments of φ_B to the auxiliary variables a, i.e.,*

$$\chi_\varphi([\![a]\!]_\mu) = \begin{cases} 1 & \text{if} \quad \exists\ b \text{ s.t.} \quad \mu(b, a) \models \varphi_B \\ 0 & \text{otherwise.} \end{cases}$$

We say that φ is an ordered positive unate formula *with respect to κ if there exists an ordering (renaming) $\kappa \colon \mathcal{I} \to \mathcal{I}$ over the index set $\mathcal{I} = \{1, \ldots, |C|\}$ such that χ_φ is ordered positive unate with respect to a', where $a' = (a'_1, \ldots, a'_{|C|}) = (a_{\kappa(1)}, \ldots, a_{\kappa(|C|)})$.*

An OPU formula φ introduces a partial order over the assignments for a that can drastically simplify the task of finding a minimal UNSAT certificate. In fact, once the set of variables a is ordered according to κ to obtain a', the assignments over a' can also be ordered based on their prefix as in Definition 5.5. A similar scenario as the one in Figure 2 will then occur, where, in this case, u represents the characteristic function χ_φ, which evaluates to 1 on a subset of vertices of the tree. It is according to this prefix-based ordering and the resulting tree that we define a "minimal" UNSAT certificate for φ.

Let us assume, for instance, that the assignment $(1, 1, 1, 0)$ is generated by SAT-SOLVE in Algorithm 4, but is found to be infeasible by $\mathcal{C}$-SOLVE. Then, an effective UNSAT certificate should maximally prune the search space of SAT-SOLVE and minimize the amount of backtracking along the branches of the tree. To find such a conflicting assignment, we must look for the closest vertex to the root of the tree, e.g., the assignment $(1, 0, 0, 0)$ in our example, such that the associated convex set of constraints is still inconsistent. The lower the vertex, the higher the number of discarded assignments from the search tree. We observe that, since the number of zeros in the assignments increases as we move backward towards the root of the tree, a minimal certificate with respect to the prefix order pictured in Figure 2 would usually produce a small clause. However, such a clause does not necessarily correspond, in general, to a minimal IIS for the associated set of convex constraints in the sense of Definition 5.1.

Based on the discussion above, finding a minimal certificate for OPU formulas amounts to looking for the longest prefix associated with a set of consistent constraints before an inconsistent constraint is reached along a path of the tree. To formalize this objective, we proceed as follows. Given an OPU formula φ with respect to an ordering κ, let $a'_\mu = (a'_{\mu,1}, \ldots, a'_{\mu,L})$ be the set of variables in $a' = (a_{\kappa(1)}, \ldots, a_{\kappa(|C|)})$ that are asserted by the valuation $\mu(b, a)$ of SAT-SOLVE, taken in the same order as in a'. We also denote as $\{(g'_{\mu,1}(x) \lhd 0), \ldots, (g'_{\mu,L}(x) \lhd 0)\}$ the set of convex constraints in φ associated with the variables a'_μ. Then, for a constant $\delta \in \mathbb{Q}^+$, we define the function $\text{ZEROPREFIX}_\delta : \mathbb{R}^L_+ \to \mathbb{N}$ as:

$$\text{ZEROPREFIX}_\delta(s_1, \ldots, s_L) = \min k \ \text{ s.t. } \sum_{i=1}^{k} |s_i| > \delta.$$

Intuitively, for small δ, zeroPrefix$_\delta$ returns the first nonzero element of the sequence $s = (s_1, \ldots, s_L)$, and practically the length of its "zero prefix." Using this function, we can then look for sequences of slack variables that maximize the number of initial elements set to zero before the first nonzero element is introduced, by casting the following optimization problem:

Problem 1.

$$\max_{\substack{s_1,\ldots,s_L \in \mathbb{R} \\ x \in \mathcal{W} \subseteq \mathbb{R}^n}} \text{zeroPrefix}_\delta(s_1, \ldots, s_L)$$
$$\text{s.t.} \qquad g'_{\mu,i}(x) \lhd s_i, \qquad i = 1, \ldots, L$$

where $\mathcal{W}$ is the domain of real variables x and $g'_{\mu,i}$, for all i, are defined as above. Problem 1 is a modified version of a conventional feasibility problem, where convex constraints are perturbed by adding slack variables s_i. By looking at the longest prefix of zero slack variables, Problem 1 is able to find the longest sequence of convex constraints that are consistent before the first "stretched" inconsistent constraint appears according to the ordering induced by κ. If the optimum prefix length is k^*, then k^* is also the length of the resulting UNSAT certificate.

A remaining drawback is the possible intractability of Problem 1 whose objective function, basically counting the number of zero elements in the prefix of a sequence, is non-convex. *It is, however, possible to still find the optimum k^* from Problem 1 using a convex program.* To state this result, at the heart of our efficient decision procedure,we consider formulas $\varphi(b, x)$ such that the domain $\mathcal{W} \in \mathbb{R}^n$ of its real variables x is bounded. Under this assumption, we are guaranteed that there is always an upper bound to the minimum sum of slack values that can make any conjunction of convex constraints in φ feasible. We can define such a bound $\bar{s}$ as follows.

Definition 5.7. *Let $\mathcal{W} \in \mathbb{R}^n$ be a bounded convex set, and $\{(g_1(x) \lhd 0), \ldots, (g_{|C|}(x) \lhd 0)\}$ the set of convex constraints in the monotone SMC formula φ. We define as $\bar{s}$ the solution of the following convex optimization problem:*

$$\max_{x \in \mathcal{W}} \min_{s_1,\ldots,s_{|C|} \in \mathbb{R}} \sum_{i=1}^{|C|} |s_i| \quad \text{s.t.} \qquad g_i(x) \lhd s_i, \quad i = 1, \ldots, |C|$$

The bound $\bar{s}$ can be easily pre-computed offline for a given φ. Then, for a given tolerance $\delta \in \mathbb{Q}^+$, we can use the following problem to find the maximum length of the zero-prefix of a sequence of slacks:

Problem 2.

$$\min_{\substack{s_1,\ldots,s_L \in \mathbb{R} \\ x \in \mathcal{W} \subseteq \mathbb{R}^n}} \sum_{i=1}^{L} |s_i|$$
$$\text{s.t.} \quad g'_{\mu,i}(x) \lhd s_i, \qquad i = 1, \ldots, L$$
$$\frac{\bar{s}}{\delta}\left(\sum_{k=1}^{i-1} |s_k|\right) \leq |s_i| \qquad i = 2, \ldots, L \qquad (7)$$

Problem 2 is a modified version of the SSF problem because of the addition of constraints (7)[3]. However, we observe that, if the problem is feasible, constraints (7) become redundant. Therefore, if the sum of slacks at optimum is zero (in practice, the condition $\sum_{i=1}^{L} |s_i| \leq \delta$ is satisfied), then μ is indeed a valid assignment. If, instead, this is not the case, constraints (7) induce an ordering over the non-zero slack variables which can be used to generate the prefix-based minimal certificate. It is therefore sufficient to solve Problem 2, as established by the following result.

[3]While constraints (7) are non-convex, they can translated into linear constraints using standard transformations dealing with the minimization of the sum of absolute values.

Algorithm 3
($\text{cStatus}, x, \varphi_{\text{ce}}$) = $\mathcal{C}$-Solve.Prefix($\mu, \mathcal{M}, \kappa, \delta$)

```
1: s* := Solve-Problem2(μ, M, κ, δ);
2: if Σ_{i=1}^{L} s*_i ≤ δ then
3:     cStatus = SAT;
4:     return (cStatus, x*, 1)
5: else
6:     cStatus = UNSAT;
7:     k* := zeroPrefix_δ(s*_1 ... s*_L);
8:     φ_ce := ⋁_{i=1}^{k*} ¬a'_{μ,i};
9:     return (cStatus, x*, φ_ce);
```

Theorem 5.8 (Prefix-Based Certificate). *Let φ be a OPU formula with respect to an ordering κ and defined over a bounded real variable domain $\mathcal{W}$. Let μ be a satisfying assignment for the propositional abstraction φ_B, $\delta \in \mathbb{Q}^+$, and $\bar{s} \in \mathbb{R}^+$, defined as in Definition 5.7, with $\bar{s} \geq \delta$. Let Problem 2 be the feasibility problem associated with μ, $\bar{s}$, and δ. Then, the following hold:*

- *If Problem 2 is feasible and x^* is the optimum for x, then $([\![b]\!]_\mu, x^*) \models \varphi$;*
- *If Problem 2 is infeasible and k^* is the minimum index such that $s^*_k > 0$, then, the following clause:*

$$\varphi_{ce} := \bigvee_{i=1}^{k^*} \neg a'_{\mu,i}, \qquad (8)$$

is an UNSAT certificate for μ which is minimal with respect to κ.

The prefix-based certificate generation procedure can then implemented as in Algorithm 3. By Theorem 5.8 we can then state the following guarantees of Algorithm 4 with the generation of UNSAT certificates in Algorithm 3.

Proposition 5.9. *Let φ be a monotone SMC formula and $\delta \in \mathbb{Q}^+$ a user-defined tolerance used in $\mathcal{C}$-Solve.Check in Algorithm 4 to accommodate numerical errors. Algorithm 4 with the certificate from Algorithm 3 is δ-complete.*

Overall, as summarized in Table 1, IIS-based certificates are generally the shortest and most effective, but also the most expensive to compute. IIS-based and SSF-based certificates can be used with any monotone SMC formula, while prefix-based certificates are the most efficient to compute for OPU formulas. As an example, OPU formulas can be used to encode the runs of a finite-state transition system, which would also form a tree, as in in Figure 2. Coupled with continuous dynamics, this pattern arises in several systems, including switched system, linear hybrid systems, piecewise affine systems, and mixed logical dynamical systems [21].

6. RESULTS

We implemented all our algorithms in the prototype solver SatEX. We use Z3 [10] as a SAT solver and Cplex [22] as

a convex optimization solver. To validate our approach, we first compare the scalability of the proposed SMC procedure with respect to state-of-art SMT and MIP solvers, such as Z3 and CPLEX, on a set of synthetically generated monotone SMC formulas. We then demonstrate the performance of SATEX and different UNSAT certificates on two hybrid system problems that both generate SMC instances: secure state estimation and robotic motion planning. All the experiments were executed on an Intel Core i7 2.5-GHz processor with 16 GB of memory. CPLEX was configured to utilize 1,2,3, or 4 processor cores.

6.1 Scalability

To test the scalability of our algorithm, we generate SMC problem instances as follows. We consider purely Boolean problem instances from the 2014 SAT competition (application track) [23] and selectively include Boolean clauses from these instances to create SMC problems with an increasing number of Boolean constraints, from 1000 to 130,000, over a maximum number of 4,288 Boolean variables. We then augment the Boolean instance with clauses of the form $\neg b_i \vee h_i(x) \leq 0$ where b_i is a pre-existing Boolean variable and h_i is a randomly generated affine function. Affine constraints are all generated in such a way that the whole SMC formula is always satisfiable. SATEX will then terminate after at most one iteration.

Figure 3 (left) reports the execution time of SATEX as the number of Boolean constraints in an SMC instance increases for a fixed number of real variables. For instances with a relatively small number of Boolean constraints (less than 15,000), MIP techniques, based on branch-and-bound and cutting plane methods, show a superior performance. However, as the number of Boolean constraints increases, the performance of SATEX, relying on SAT solving, exceeds the one of MIP techniques by 4-5 orders of magnitude in execution time. The performance gap between the lazy procedure of SATEX and Z3 is also observed to increase with the number of Boolean constraints, and reach more than one order of magnitude. On the other hand, when the number of continuous variables in the affine constraints increases, as shown on the right side of Figure 3, Z3 reaches a 600-s timeout on problem instances with more than 1500 continuous variables, while optimization-based algorithms show the expected polynomial degradation, with SATEX running approximately twice as faster as MIP.

Next, we consider SMC formulas that are certified to be unsatisfiable, since they are directly created using UNSAT Boolean instances from the SAT 2014 competition, augmented with linear constraints as above. As shown in Figure 4, again, when relying on SAT solving to detect unsatisfiability, SATEX runs faster by two orders of magnitude with respect to MIP based techniques. Its performance is, in this case, comparable with the one of Z3.

6.2 Application to Secure State Estimation

Given a set of p sensor measurements $Y_1, Y_2, \ldots, Y_p$ out of a linear dynamical system, the secure state estimation problem consists in reconstructing the state x of the dynamical system even if up to k sensors are maliciously corrupted [14]. The sensor measurements are a function of the system state, where

$$Y_i = \begin{cases} H_i x & \text{if sensor } i \text{ is attack-free} \\ H_i x + \alpha_i & \text{if sensor } i \text{ is under attack} \end{cases}$$

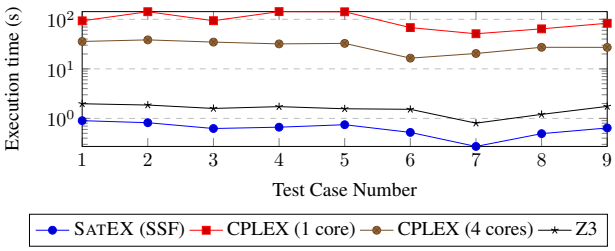

Figure 4: Execution time on UNSAT SMC instances due to UNSAT Boolean constraints: the number of Boolean clauses varies from 225 to 960, while the number of real variables is fixed to 500.

and α_i models the attack injection. It is possible to encode the secure state estimation problem as an SMC instance by introducing indicator variables b_i that are assigned to 1 if and only if the sensor is attacked. We therefore obtain the following monotone SMC formula:

$$\varphi ::= \left(\sum_{i=1}^{p} b_i \leq k\right) \wedge \bigwedge_{i=1}^{p} \left(\neg b_i \rightarrow (\|Y_i - H_i x\|_2^2 \leq \nu)\right)$$

where the first constraint is a pseudo-Boolean predicate that requires that no more than k sensors be under attack, while the other constraints establish that the state x is linearly related with the measurements in the case of attack-free sensors, except for an error bounded by $\nu \in \mathbb{R}^+$.

As shown in Figure 5, SATEX outperforms the MIP solver by up to 1 order of magnitude as the number of sensors (hence the number of Boolean variables and constraints) increases. Moreover, IIS-based certificates are often not better than SSF-based certificates, even if they are minimal, because of the cost paid for constructing them. In all our benchmarks, Z3 exceeds the 600-s timeout, possibly because of the longer run times of the nonlinear real arithmetic theory required by the quadratic constraints.

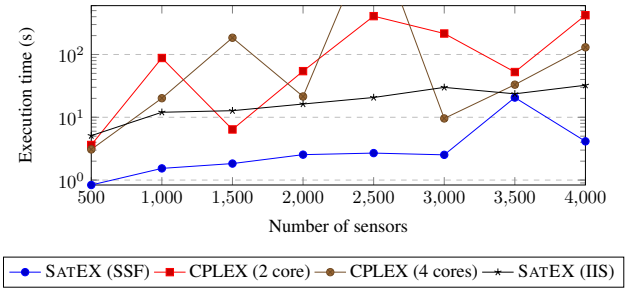

Figure 5: Execution time on instances of the secure state estimation problem when the number of sensors increase. Z3 exceeds a 600-s timeout in all benchmarks.

6.3 Application to Motion Planning

The reach-avoid problem examined in Section 2 reduces to an OPU formula, as suggested by the ordering of the Boolean variables associated with the different regions according to the transition system in Figure 1. We can, therefore, exploit our results on prefix-based UNSAT certificates.

Figure 6 shows the runtime performance of SATEX with respect to a MIP solver on instances of the motion planning problem. We assume that we operate with the linearized dynamics of a quadrotor (having 14 continuous states) moving

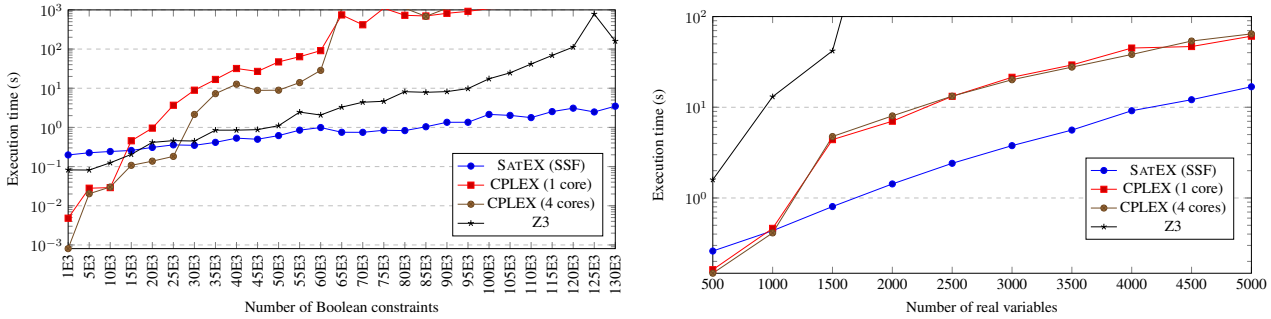

Figure 3: Execution time on SMC problem instances, when the number of Boolean constraints increases for a fixed number of 100 real variables (left side) and when the number of real variables increases for a fixed number of 7000 Boolean constraints (right side).

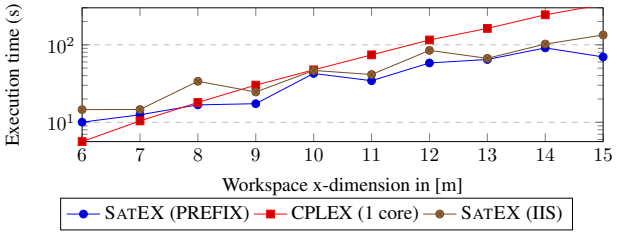

Figure 6: Execution time on a set of SMC instances for the motion planning problem as the size of the workspace increases.

in a 3-dimensional workspace. Moreover, we partition the workspace into cubes of size $1m \times 1m \times 1m$ and randomly select some of them to be obstacles. We then keep fixed to 4 m the workspace width and height and let its length increase (along the x axis). This translates into increasing both the number of Boolean and continuous variables in φ, since L must also increase in order to reach the target. Consistently with our previous observations, increasing the number of Boolean variables directly maps into a larger performance gap associated with prefix-based UNSAT certificates, which outperform both the IIS-based and MIP-based approaches.

7. CONCLUSIONS

We demonstrated a procedure for determining the satisfiability of two special, yet common, types of logic formulas over Boolean and convex constraints. By leveraging the strengths of both SAT solving and convex programming as well as efficient conflict-driven learning strategies, our approach outperforms state-of-the-art SMT and MICP solvers on problems with complex Boolean structure and a large number of real variables. The proposed satisfiability modulo convex optimization scheme can then be used to build effective and scalable decision procedures for problems in hybrid system verification and control.

Acknowledgments

This work was partially sponsored by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, by the NSF project ExCAPE: Expeditions in Computer Augmented Program Engineering, and by the NSF award 1239085.

8. REFERENCES

[1] J. N. Hooker, "Logic, optimization, and constraint programming," *INFORMS Journal on Computing*, vol. 14, no. 4, pp. 295–321, 2002.

[2] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, *Satisfiability Modulo Theories, Chapter in Handbook of Satisfiability*. IOS Press, 2009.

[3] S. Ratschan, "Efficient solving of quantified inequality constraints over the real numbers," *ACM Trans. Comput. Logic*, vol. 7, no. 4, pp. 723–748, 2006.

[4] S. Gao, J. Avigad, and E. M. Clarke, "δ-complete decision procedures for satisfiability over the reals," in *Proc. Int. Joint Conf. Automated Reasoning*, 2012, pp. 286–300.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univesity Press, 2004.

[6] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T)," *J. ACM*, vol. 53, no. 6, pp. 937–977, Nov. 2006.

[7] A. Cimatti *et al.*, "Satisfiability modulo the theory of costs: Foundations and applications," in *Proc. TACAS*, 2010, pp. 99–113.

[8] Y. Li *et al.*, "Symbolic optimization with SMT solvers," in *ACM SIGPLAN Notices*, vol. 49, no. 1, 2014, pp. 607–618.

[9] A. Bauer, M. Pister, and M. Tautschnig, "Tool-support for the analysis of hybrid systems and models," in *Proc. of DATE*, 2007.

[10] L. De Moura and N. Björner, "Z3: An efficient SMT solver," in *Proc. Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems*, 2008, pp. 337–340.

[11] M. Franzle *et al.*, "Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure," in *JSAT Special Issue on SAT/CP Integration*, 2007.

[12] S. Gao, S. Kong, and E. M. Clarke, "dReal: An SMT solver for nonlinear theories over the reals," 2013, vol. 7898, pp. 208–214.

[13] P. Nuzzo *et al.*, "CalCS: SMT solving for non-linear convex constraints," in *Proc. Formal Methods in Computer-Aided Design*, Oct. 2010, pp. 71–79.

[14] Y. Shoukry *et al.*, "Sound and complete state estimation for linear dynamical systems under sensor attack using satisfiability modulo theory solving," in *Proc. American Control Conference*, 2015, pp. 3818–3823.

[15] Y. Shoukry *et al.*, "Scalable lazy SMT-based motion planning," in *Proc. Int. Conf. Decision and Control*, 2016, pp. 6683–6688.

[16] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI Communications*, no. Preprint, pp. 1–12.

[17] A. Pnueli, "The temporal logic of programs," in *FOCS*, 1977, pp. 46–57.

[18] M. Grant, S. Boyd, and Y. Ye, "Disciplined convex programming," in *Global optimization*. Springer, 2006, pp. 155–210.

[19] Y. Shoukry, P. Nuzzo, A. Sangiovanni-Vincentelli, S. Seshia, G. Pappas, and P. Tabuada, "SMC: Satisfiability modulo convex optimization," *ArXiv e-prints*, 2017.

[20] J. W. Chinneck and E. W. Dravnieks, "Locating minimal infeasible constraint sets in linear programs," *ORSA Journal on Computing*, vol. 3, no. 2, pp. 157–168, 1991.

[21] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, 1999.

[22] (2012, Feb.) IBM ILOG CPLEX Optimizer. [Online]. Available: www.ibm.com/software/integration/optimization/cplex-optimizer/

[23] "The international SAT competitions web page."

http://www.satcompetition.org/, accessed: 2016-10-01.