This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

INVITED PAPER

# SMC: Satisfiability Modulo Convex Programming

By Yasser Shoukry, *Member IEEE*, Pierluigi Nuzzo, *Member IEEE*, Alberto L. Sangiovanni-Vincentelli, *Fellow IEEE*, Sanjit A. Seshia, *Fellow IEEE*, George J. Pappas, *Fellow IEEE*, and Paulo Tabuada, *Fellow IEEE*

**ABSTRACT** | The design of cyber–physical systems (CPSs) requires methods and tools that can efficiently reason about the interaction between discrete models, e.g., representing the behaviors of "cyber" components, and continuous models of physical processes. Boolean methods such as satisfiability (SAT) solving are successful in tackling large combinatorial search problems for the design and verification of hardware and software components. On the other hand, problems in control, communications, signal processing, and machine learning often rely on convex programming as a powerful solution engine. However, despite their strengths, neither approach would work in isolation for CPSs. In this paper, we present a new satisfiability modulo convex programming (SMC) framework that integrates SAT solving and convex optimization to efficiently reason about Boolean and convex constraints at the same time. We exploit the properties of a class of logic formulas over Boolean and nonlinear real predicates, termed monotone satisfiability modulo convex formulas, whose satisfiability can be checked via a finite number of convex programs. Following the lazy satisfiability modulo theory (SMT) paradigm, we develop a new decision procedure for monotone SMC formulas, which coordinates SAT solving and convex programming to provide a satisfying assignment or determine that the formula is unsatisfiable. A key step in our coordination scheme is the efficient generation of succinct infeasibility proofs for inconsistent constraints that can support conflict-driven learning and accelerate the search. We demonstrate our approach on different CPS design problems, including spacecraft docking mission control, robotic motion planning, and secure state estimation. We show that SMC can handle more complex problem instances than state-of-the-art alternative techniques based on SMT solving and mixed integer convex programming.

**KEYWORDS** | Cyber–physical systems; system-level design; system verification

## I. INTRODUCTION

Cyber–physical systems (CPSs) result from the integration of computation and communication with physical processes, and their behaviors are defined by both cyber and physical parts of the system [1]. CPSs subject to tight safety, reliability, security, and cost requirements are increasingly being deployed in several areas, including transportation, health-care, and infrastructure. These systems would dramatically benefit from algorithmic techniques to enhance design quality and productivity and enable autonomy under strong guarantees of correctness and dependability [2]–[7]. However, their complexity and heterogeneity pose several challenges to design automation.

Because of their heterogeneous nature, analysis and design of CPSs increasingly require methods and tools that can efficiently reason about the interaction between discrete models, e.g., used to describe embedded software components, and continuous models used to describe physical processes. In this respect, a central difficulty is the very different nature of the tools used to analyze continuous dynamics (e.g., real analysis) and discrete dynamics (e.g., combinatorics) as well as solve constraint satisfaction

Manuscript received September 4, 2017; revised March 27, 2018; accepted May 19, 2018. *(Corresponding author: Yasser Shoukry.)*

**Y. Shoukry** was with the University of California at Los Angeles, Los Angeles, CA USA, and also with the University of California at Berkeley, Berkeley, CA, USA. He is now with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20740 USA (e-mail: yshoukry@ece.umd.edu).

**P. Nuzzo** is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA.

**A. L. Sangiovanni-Vincentelli** is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA.

**S. A. Seshia** is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA.

**G. J. Pappas** is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA.

**P. Tabuada** is with the Electrical and Computer Engineering Department, University of California at Los Angeles, Los Angeles, CA 90095 USA.

Digital Object Identifier 10.1109/JPROC.2018.2849003

0018-9219 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



problems involving continuous and discrete variables. This difficulty is exacerbated by complex, high-dimensional systems, where a vast discrete/continuous space must be searched under constraints that are often nonlinear. Methods that substantially rely on discrete system abstractions, often obtained by partitioning the continuous state space into polytopes, and automata-theoretic approaches [7]–[11] are subject to the *curse of dimensionality* and become usually impractical for systems with more than five continuous states [12].

Boolean methods such as satisfiability (SAT) solving have been successful in tackling large combinatorial search problems for the design and verification of hardware and software systems [13]. SAT solvers are the reasoning engine behind commercial verification and testing tools in the electronic design automation industry. SAT has also been used in tools for software verification and debugging, for example, industrial verification of device drivers and static analysis. The formulation of new SAT encodings has made SAT solvers powerful engines for solving Boolean or discrete constraint satisfaction problems from an increasingly wider range of applications, from routing circuits to validating software models, from scheduling and planning in artificial intelligence to synthesizing consistent network configurations.

On the other hand, problems in control, communications, signal processing, data analysis and modeling, and machine learning often rely on convex programming (CP) as a powerful solution engine [14]. Convex optimization problems can be solved very efficiently today, based on a mature theory. The solution methods have proved to be reliable enough to be embedded in computer-aided design or analysis tools, or even in real-time reactive or automatic control systems. Moreover, whenever the original problem is not convex, convex problems can still provide the starting point for other local optimization methods, or computationally efficient approximations via constraint or Lagrangian relaxations. However, despite their strengths, neither SAT solving nor CP would be effective in CPS design, if used in isolation. We need methods and computational tools that blend concepts from both of them.

Satisfiability modulo theory (SMT) solving has emerged over the years as a paradigm for extending the reasoning capabilities of SAT solvers to address richer constraint satisfaction problems. Modern SMT solvers [15] can efficiently find satisfying valuations of very large 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. SMT solvers usually implement constraint programming techniques [16] and logic-based inference methods such as domain reduction and constraint propagation to accelerate the search. However, while SMT solving for generic nonlinear theories over the reals is undecidable [17], [18], constraint programming techniques can still be computationally expensive for decidable instances with large and expressive formulas. Algorithms and tools that can address combinations of Boolean constraints and useful fragments of the nonlinear theories with solid guarantees of correctness and scalability are highly needed.

Optimization-based approaches, such as mixed integer linear programming (MILP) and mixed integer convex programming (MICP), have shown to be capable of solving for discrete and continuous convex constraints at the same time while striking a good balance between expressiveness and computational efficiency [19]–[21]. MICP-based approaches encode a logic combination of Boolean and convex constraints into a conjunction of mixed integer convex constraints and solve it by leveraging numerical algorithms for convex optimization in combination with branch-and-bound and cutting-plane methods. State-of-the-art solvers show superior empirical performance when handling large numbers of hybrid constraints and variables. They, however, tend to become inefficient when the Boolean structure of the problem becomes complex. Moreover, 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 [16], which may eventually affect the correctness of the solution.

In this paper, we rethink the connection between Boolean methods and convex optimization toward a novel, scalable framework for reasoning about the combination of discrete and continuous dynamics that can address the complexity of CPS applications. As pictorially sketched in Fig. 1, SAT solving and CP 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 and have been the subject of increasing research activity [16], devising a robust and widely acceptable scheme combining the advantages of both approaches is still largely an open issue.

We address this challenge by focusing on the satisfiability problem for a class of formulas over Boolean variables and convex constraints. We 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 can

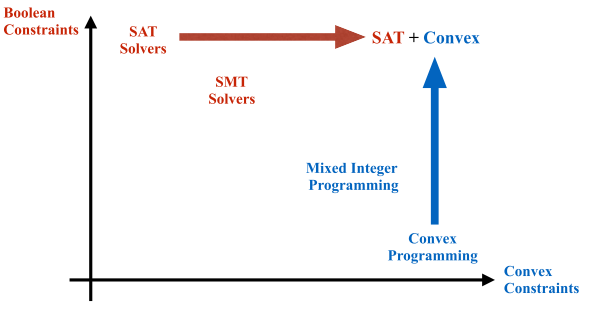



**Fig. 1.** ***Pictorial representation of the capabilities of constraint programming and optimization techniques as well as the satisfiability modulo convex programming approach.***

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



be solved via a finite number of convex programs. For monotone SMC formulas, we develop a new procedure, which we call satisfiability modulo convex programming, 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 [15], [22], a classic SAT solving algorithm 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. By leveraging the efficiency and formal guarantees of state-of-the-art constraint solving algorithms in both the Boolean and convex analysis domains, SMC strives to alleviate the scalability issues associated with the discretization of the continuous variables.

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. We therefore 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 subclass of monotone SMC formulas, namely prefix-ordered monotone SMC (POM) formulas, that present additional monotonicity properties. Since monotone SMC and POM 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.

The rest of the paper is organized as follows. After an overview of the related work in Section II, Section III introduces a representative set of CPS design problems, which will be used throughout the paper to illustrate the relevance of our approach. Section IV presents the formal definition of monotone SMC formulas and their properties. Further, it details how the reference design problems in Section III can be encoded into satisfiability problems for monotone SMC formulas. Section V describes the overall SMC solution strategy, while Section VI develops algorithms to find compact infeasibility certificates. Finally, Section VII discusses the validation of our techniques and their application to the reference design problems, while Section VIII concludes with a summary of our work.

## II. RELATED WORK

Our algorithm follows the lazy SMT solving paradigm [15], where a classic David–Putnam–Logemann–Loveland (DPLL)-style SAT solving algorithm interacts with a theory solver [22]. An SMT instance is a formula in first-order logic, where some function and predicate symbols have additional interpretations related to specific theories, and SMT is the problem of determining whether such a formula is satisfiable. Our focus is, therefore, on feasibility problems and on leveraging optimization methods to accelerate the search for satisfying assignments. In this respect, our work differs from other research efforts such as the "optimization modulo theories" [23] or "symbolic optimization" [24] approaches, which propose SMT-based techniques to solve optimization problems.

The ABSOLVER tool [25] 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 [26] solver can also provide support for nonlinear polynomial arithmetic, while possibly incurring high computational costs [27]. The iSAT algorithm builds on a unification of SAT-solving and interval constraint propagation (ICP) [28] to efficiently address arbitrary smooth, possibly transcendental, functions. The integration of SAT solving with ICP is also used in dREAL [29] to build a $\delta$-complete decision procedure which solves SMT problems over the reals with nonlinear functions, such as polynomials, sine, exponentiation, or logarithms, but with limited support for logic combinations of Boolean and real constraints. Contrary to the previous approaches, 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 [30]. CalCS also leverages conservative approximations of reverse (negated) convex constraints to implement a semi-decision procedure for nonmonotone convex SMT formulas, and has been used on benchmarks from bounded model checking of hybrid automata and static analysis of floating-point software. 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 control. For SMC formulas, we provide formal correctness guarantees for our algorithms in terms of $\delta$-completeness [18]. 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 duality theory and the sensitivity of the objective of a convex optimization problem to its constraints.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



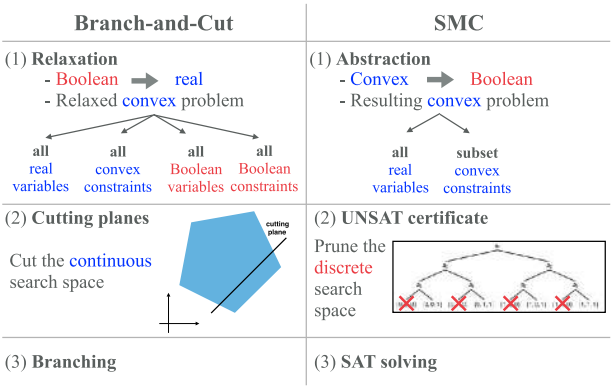



**Fig. 2.** ***Comparison between mixed integer convex programming (MICP) based techniques and the SMC approach.***

We have recently developed specialized SMT-based algorithms for applications in secure state estimation, Imhotep-SMT [31]–[34], and robotic motion planning [35], [36]. We show that the approach detailed in this paper subsumes these results. A preliminary version of the results in this paper appeared in our previous publications [36], [37], without the proofs of the formal guarantees of our algorithms. In this paper, we discuss and prove in detail all the results used in our previous work and demonstrate our approach on a new example, an Autonomous spacecraft Rendezvous, Proximity Operations, and Docking (ARPOD) problem, which has been recently proposed as an exemplar benchmark [38] for the development and validation of hybrid systems analysis and control techniques.

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. A comparison between the solution strategies adopted by MICP and SMC is suggested in Fig. 2. SMC exploits abstraction of convex constraints and conflict-driven learning, together with the structure of monotone SMC formulas, to directly reduce the search space and decompose the original problem into a sequence of simpler convex programs. Conversely, MILP-based approaches leverage branching and cutting planes to generate a sequence of simpler continuous or Lagrangean relaxations of the original problem and its Boolean constraints. There are analogies in the way progress is made in the two approaches. Cutting planes accelerate the search by cutting the continuous search space. Similarly, UNSAT certificates accelerate the search by pruning the discrete search space. Branching is used in MICP solvers to generate increasingly better relaxations. Similarly, SAT solving is used in SMC to generate increasingly better Boolean assignments. Both techniques decompose the solution of a complex hybrid problem into solving a sequence of simpler ones. However, the number of variables and constraints in each convex problem instance is usually much smaller in the SMC approach, while it may become prohibitively high in MICP-based approaches, sometimes on the order of the number of all the Boolean and real variables and constraints of the original problem. Overall, 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.

## III. MOTIVATING EXAMPLES

We introduce a representative set of CPS design problems that will be used to illustrate the approach in this paper, showing how estimation and control design problems arising in different contexts can be formulated and efficiently solved within the SMC framework.

### A. Spacecraft Docking Mission

The realization of autonomous spacecraft that can operate independently of human control across a number of commercial, civil, and military missions and under a wide variety of operating conditions has attracted significant attention in recent years. A key challenge in this context is the autonomous navigation and control of the motion of one spacecraft relative to another spacecraft, including docking of two spacecraft on-orbit. This is an indispensable component of several missions, such as manned spaceflight involving the on-orbit transfer of personnel and resupply missions providing material for on-orbit personnel, assembly, servicing, and repair. An ARPOD problem has been recently proposed as an exemplar benchmark [38] for the development and validation of hybrid systems analysis and control techniques.

As shown in Fig. 3, an ARPOD mission typically consists of a sequence of phases based on the distance between a *target* spacecraft, which is passive or station-keeping, and a *chaser* spacecraft, which actively controls the maneuvers. In the *rendezvous* phase, the chaser approaches the target, typically in a range of 10 km to 100 m of separation. The *docking* phase describes the final maneuvers executed to engage the docking ports and covers the range from 100 m to 0 m. Finally, the *docked* phase describes the control of the rigidly attached spacecraft pair. In all phases, the goal is to minimize the amount of fuel or propellant consumed, since this directly impacts the spacecraft lifetime.

We assume that the chaser must reach a space station module (target) and transport it to an assembly location

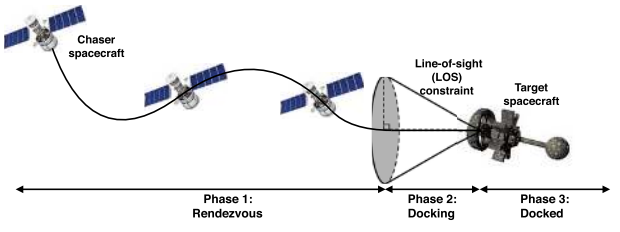



**Fig. 3.** ***Schematic of a spacecraft docking maneuver subject to LOS constraints with snapshots of the chaser spacecraft at three different time instants.***

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



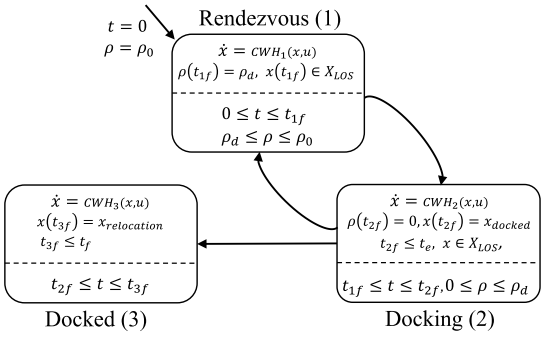



**Fig. 4.** ***Hybrid system representation of the ARPOD mission.***

before a predefined mission end time. The mission lends itself to a natural description in terms of a hybrid system, as shown in Fig. 4. For a fixed horizon $L$, the controller design problem translates into finding a system trajectory (sequence of states of length $L$) that brings the chaser from the initial point to the target and then to the assembly location, while satisfying a set of mission constraints.

The mission constraints consist of a combination of Boolean and continuous constraints that are convex. We assume that the continuous dynamics in each hybrid system mode $i$ are discretized based on the sampling time $t_s^i$. By leveraging encoding techniques from bounded model checking [39], a set of *Boolean constraints* can capture the transition relation between modes. We introduce a Boolean variable $b_k^i$ for each mode $i$ and time $k$ such that $b_k^i$ is true if and only if the system is in mode $i$ at time $k$. We require that the system be only in one mode at each time. Moreover, if the system is in mode $i$ at time $k$, it can only stay in mode $i$ or transition to a direct successor mode at time $k+1$.

Additionally, in each mode, the continuous state of the system, representing the position and velocity of the chaser, is subject to a set of linear, time-invariant, difference equations describing the translational motion of the chaser before and after docking in a suitable coordinate frame. The control input at any point in time is also bounded by the maximum thrust that can be produced in each of the axial directions. Both the system dynamics and the control bounds can be represented by sets of linear *constraints on continuous variables* that must hold in all modes.

Finally, there are continuous constraints that are specific to each phase of the mission. The chaser begins its maneuvers in Phase 1 while its relative displacement $\rho$ from the target satisfies $\rho_0 \geq \rho \geq \rho_d$, where $\rho_0$ is the initial displacement, within 10 km, and $\rho_d = 100$ m. Moreover, Phase 1 must end in the line-of-sight (LOS) of the sensors available for docking at a distance equal to $\rho_d$.

After the chaser moves to a position for which $\rho < \rho_d$, Phase 2 is initiated, in which the chaser spacecraft attempts to reduce $\rho$ to zero while remaining in the LOS region and maintaining a slow velocity to reduce impact forces upon docking. The sensing and control frequency may increase in this phase, which translates into a smaller sampling time and a different discretization of the dynamics. The LOS constraint can be captured as either a nonlinear cone constraint, as in Fig. 3, or a linear pyramid constraint. The upper bound on the velocity can be specified by a nonlinear $\ell_2$-norm constraint or a linear $\ell_\infty$-norm constraint. In all cases, these constraints are convex. In Phase 2 the chaser must also dock to the target before the eclipse time.

Once the chaser spacecraft docks (i.e., $\rho = 0$ m), the spacecraft pair enters Phase 3, where the joint assembly must move to the relocation position. The constraints in this phase require that the assembly reach the relocation spot by the mission end time.

## B. Robotic Motion Planning

While seemingly addressing a purely continuous problem, developing algorithmic techniques for robotic motion planning actually requires reasoning about the tight integration of discrete abstractions (as in *task planning*) with continuous motions (*motion planning*) [40]. Task planning relies on high-level specifications of temporal goals that are most conveniently captured by logics such as linear temporal logic (LTL) [41]. Motion planning deals, instead, 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. This is, indeed, possible within the SMC framework.

For simplicity, we discuss the basic reach-avoid problem, which is the foundation of more complex motion planning problems [35]. However, our approach extends to multirobot motion planning from generic LTL specifications [36]. 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. 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. Further, as conveniently done in the context of task planning, we consider an abstraction of the workspace in terms of a set of regions, as shown in Fig. 5 (left). We assume that regions and obstacles are described by polyhedra and captured by linear constraints in 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 length $L$ of regions (discrete plan) that brings the robot from the initial point to the target and is compatible with the continuous dynamics.

As in the ARPOD mission example, the problem constraints consist of a combination of Boolean and convex constraints. The adjacency relation between regions can be captured via a transition system as in Fig. 5 (right).

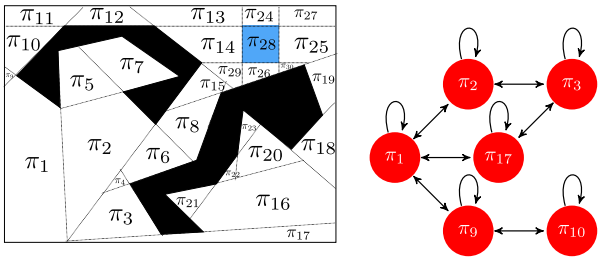



**Fig. 5.** ***Discretization of the workspace for the motion planning problem (left), and transition system describing the adjacency relation between regions in the workspace (right).***

A valid trajectory for the robot can then be represented by a run of the transition system. Let $b_i^k$ be a Boolean variable that evaluates to true if and only if the robot is in region $i$ at time $k$. We can then encode the transition relation between regions via a set of *Boolean constraints*. *Convex constraints* can instead be used to capture the robot dynamics, the upper bound on the feasible magnitude (e.g., $\ell_2$- or $\ell_\infty$-norm) of the control input at time $k$, and the constraint that the state at time $k$ must belong to region $i$ if $b_i^k$ is true.

### C. Secure State Estimation

The detection and mitigation of attacks on CPSs is a problem of increasing importance. In these systems, the increased sophistication often comes at the expense of increased vulnerability and security weaknesses. An important scenario is posed by a malicious adversary that can arbitrarily corrupt the measurements of a subset of sensors in the system. Because sensor measurements are used to generate control commands, corrupted measurements can lead to corrupted commands, thus critically affecting the physical process under control. One way of counteracting these attacks is to attempt at estimating the state of the underlying physical system from a set of noisy and adversarially corrupted measurements, so that it can be used by the controller. We call this problem secure state estimation [31].

Even if the physical system has only continuous dynamics, secure state estimation is intrinsically a combinatorial problem, which has been traditionally addressed either by brute force search, suffering from scalability issues, or via convex relaxations, using algorithms that can terminate in polynomial time but are not necessarily sound. However, as for the problems in Section III-A and III-B, this problem is also amenable to an exact formulation and efficient solution techniques within the SMC framework [37].

We focus on linear dynamical systems and model the attack as a sparse vector added to the measurement vector. The entries corresponding to unattacked sensors are null while sensors under attack are corrupted by nonzero signals. We make no assumptions regarding the magnitude, statistical description, or temporal evolution of the attack vector. Given a set of $p$ sensor measurements taken over a time window, the secure state estimation problem consists of reconstructing the state of the dynamical system even if up to $k$ sensors ( $k \leq p$ ) are maliciously corrupted.

As in the other examples in this section, it is possible to express the problem constraints via a combination of Boolean and convex constraints. The available information on the sensors under attacks can be encoded by introducing, for each sensor $i$, a Boolean variable $b_i$ which evaluates to true if and only if the sensor is attacked. *Boolean constraints* can then be formulated to require that no more than $k$ sensors be under attack. *Convex constraints* can be added to require that the state be linearly related with the measurements in the case of attack-free sensors, except for a bounded error accounting for modeling and measurement errors.

Overall, while relating to substantially different contexts, the estimation and control design problems in this section share the same underlying structure. In Section IV, we show that these problems can all be captured as satisfiability problems for a conjunction of logic clauses, possibly including pseudo-Boolean predicates (e.g., cardinality constraints), and where some of the literals are convex constraints. We call such a formula a *monotone SMC formula*, since none of the convex constraints are negated. In the following, we start by defining the syntax and semantics of SMC formulas and then detail the translation of the design problems in this section into monotone SMC formulas.

## IV. SATISFIABILITY MODULO CONVEX FORMULAS

### A. Notation

We denote with $b = (b_1, b_2, \ldots, b_m)$ the set of Boolean variables in a formula, with $b_i \in \mathbb{B}$, and with $x = (x_1, x_2, \ldots, x_n)$ 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 $\varphi$ is defined. A valuation $\mu$ is a function that associates each variable in $b$ and $x$, respectively, to a truth value in $\mathbb{B}$ and a real value in $\mathbb{R}$. $\top$ and $\bot$ denote, respectively, the Boolean values *true* and *false*, while $[\![b, x]\!]_\mu \in \mathbb{B}^m \times \mathbb{R}^n$ denotes the values assigned to each variable in $b$ and $x$ by $\mu$. We also say that variable $b_i$ is asserted if $[\![b_i]\!]_\mu = \top$.

A set $\mathcal{C}$ is convex if the line segment between any two points in $\mathcal{C}$ lies in $\mathcal{C}$, i.e., if for any $x_1, x_2 \in \mathcal{C}$ and any $\theta$ with $0 \leq \theta \leq 1$, we have $\theta x_1 + (1-\theta)x_2 \in \mathcal{C}$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is termed *convex* if its domain $\mathcal{D}$ is a convex set and if for all $x, y \in \mathcal{D}$, and $\theta$ with $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y). \tag{1}$$

Geometrically, this inequality means that the *chord* from $x$ to $y$ lies above the graph of $f$ [14]. As a special case, when (1) always holds as an equality, then $f$ is *affine*. All linear functions are also affine, hence convex.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



A function $f : \mathbb{R}^n \to \mathbb{R}$ with domain $\mathcal{D}$ is *closed* when, for all $\alpha \in \mathbb{R}$, the sublevel set $\{x \in \mathcal{D} | f(x) \leq \alpha\}$ is closed or, equivalently, the set $\{(x, s) \in \mathbb{R}^{n+1} | x \in \mathcal{D}, f(x) \leq s\}$ is closed.

A *convex constraint* is a constraint of one of the following forms: $f(x) < 0$, $f(x) \leq 0$, 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.[1] We further denote the negation of a convex constraint, expressed in the form $f(x) \geq 0$ ($f(x) > 0$), as a *reverse convex* constraint. A reverse convex constraint is, in general, nonconvex and so is its satisfying set.

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

## B. Syntax and Semantics

We represent SMT formulas over convex constraints to be quantifier-free formulas in conjunctive normal form, with atomic propositions ranging over propositional variables and arithmetic constraints on (closed) convex functions [30]. We call this set of formulas satisfiability modulo convex (SMC) formulas.

**Definition 4.1 (SMC Formulas):** An SMC formula is any formula that can be represented using the following syntax:

$$\begin{aligned} formula &::= \{clause \wedge\}^* clause \\ clause &::= (\{literal \vee\}^* literal) | \\ &\quad\ pBool_predicate \\ literal &::= bool_var | \neg bool_var | \top | \bot | \\ &\quad\ conv_constraint | \\ &\quad\ \neg conv_constraint \\ conv_constraint &::= equality | inequality \\ equality &::= affine_function = 0 \\ inequality &::= convex_function\ relation\ 0 \\ relation &::= < | \leq \end{aligned} \tag{2}$$

In the grammar above, *bool_var* denotes a Boolean variable, *pBool_predicate* a pseudo-Boolean predicate, and *affine_function* and *convex_function* denote affine and convex functions, respectively. We further assume that the convex functions and their domains are closed. We rely on the disciplined convex programming approach [14], [42] 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 $\mu$ (i.e., $[\![b, x]\!]_\mu \in \mathbb{B}^m \times \mathbb{R}^n$). A formula $\varphi$ is satisfied by a valuation $\mu$ ($\mu \models \varphi$) 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 $[\![l]\!]_\mu = \top$. The equality constraint $h(x) = 0$ is satisfied when the equality $h([\![x]\!]_\mu) = 0$ holds between the real numbers $h([\![x]\!]_\mu)$ and 0. The same notion of satisfaction applies to the inequalities $f(x) < 0$ and $f(x) \leq 0$, using the standard interpretation of the ordering relations over the reals.

SMC formulas require, in general, the solution of nonconvex feasibility problems to find a *model*, i.e., a satisfying assignment. To see this, consider, for instance, the formula:

$$b \wedge (x_1^2 + x_2^2 - 2x_1 - 1 \leq 0) \wedge \left(\neg b \vee (x_1^2 + x_2^2 - 1 \geq 0)\right) \tag{3}$$

where the constraint $(x_1^2 + x_2^2 - 1 \geq 0)$, defining the feasible set, is nonconvex. We are, however, interested in formulas for which a model can always be found by only solving one (or more) convex feasibility problems. This is the case for monotone SMC formulas, defined as follows.

**Definition 4.2 (Monotone SMC Formula):** A monotone SMC formula is any formula that can be represented using the following syntax:

$$\begin{aligned} formula &::= \{clause \wedge\}^* clause \\ clause &::= (\{literal \vee\}^* literal) | \\ &\quad\ pBool_predicate \\ literal &::= bool_var | \neg bool_var | \top | \bot | \\ &\quad\ conv_constraint \\ conv_constraint &::= equality | inequality \\ equality &::= affine_function = 0 \\ inequality &::= convex_function\ relation\ 0 \\ relation &::= < | \leq \end{aligned} \tag{4}$$

Monotone SMC formulas can only admit convex constraints as theory atoms. Differently from generic (nonmonotone) SMC formulas, reverse convex constraints, such as $(x_1^2 + x_2^2 - 1 \geq 0)$ in (3), are not allowed. The monotonicity property is key to guarantee that a model can always be found by solving one (or more) optimization problems that are convex, as we further discuss below.

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 bound on the numerical error can be made very small. Therefore, to provide

[1] 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 $(\tilde{g}(x) \lhd 0) = (g(x) \lhd 0) \wedge (d(x) \leq 0)$.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



correctness guarantees for our algorithms, we resort to notions of $\delta$-satisfaction and $\delta$-completeness similar to the ones previously proposed by Gao et al. [18], which we define below for generic SMC formulas.

**Definition 4.3 ($\delta$-Relaxation):** Given an SMC formula $\varphi$, let $|C|$ be the number of convex constraints in $\varphi$ and $\delta \in \mathbb{Q}^+ \cup \{0\}$ any non-negative rational number $\delta$. We define a $\delta$-relaxation of $\varphi$ as a formula obtained by replacing any convex constraints of the forms $f_i(x) < 0$, $f_i(x) \leq 0$, and $h_j(x) = 0$ in $\varphi$ with perturbed versions $f_i(x) \leq \delta_i$ and $|h_j(x)| \leq \delta_j$, where $\delta_k \in \mathbb{Q}^+ \cup \{0\}$ for all $k \in \{1, \ldots, |C|\}$, and such that $\sum_{k=1}^{|C|} \delta_k \leq \delta$.

**Definition 4.4 ($\delta$-Satisfaction):** Given an SMC formula $\varphi$ and $\delta \in \mathbb{Q}^+$, we say that $\varphi$ is $\delta$-SAT if there exists a $\delta$-relaxation of $\varphi$ that is satisfiable.

We simply say that $\varphi$ is SAT when there is no ambiguity about the choice of $\delta$. If $\varphi$ is satisfiable, then any $\delta$-relaxation of $\varphi$ is satisfiable for all $\delta \in \mathbb{Q}^+$. The opposite is, however, not true. In fact, depending on the value of $\delta$, $\varphi$ and its $\delta$-relaxation 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 $\delta$- *complete* if it can correctly solve the satisfiability problem for an SMC formula in the sense of Definition 4.4.

We can determine the $\delta$-satisfaction of an SMC formula and the $\delta$-completeness of our algorithms, in that we leverage results from convex optimization theory and state-of-the-art convex optimization algorithms [14] that can control the suboptimality of a solution, and therefore the accuracy of the result. In particular, the following proposition is a reformulation of a classical result on the convergence of the *projected gradient method* [43], [44] for optimization of a convex function over a convex, closed, and nonempty set.

**Proposition 4.5:** Given the convex optimization problem

$$\min \quad f_0(x) \quad \text{s.t.} \quad x \in \mathcal{C} \tag{5}$$

where $f_0 : \mathbb{R}^n \to \mathbb{R}$ is convex, $\mathcal{C} \subseteq \mathbb{R}^n$ is a closed, convex, nonempty set, and $f_0$ is continuously differentiable on $\mathcal{C}$, that achieves its optimum value $p^*$ on some $x^* \in C$, i.e., $f(x^*) = p^*$. If $x^{(k)}$, $k = 0, 1, \ldots$, is a sequence of iterates generated by a projected gradient method with appropriate step size selection [43], [44], then $x^{(k)}$ is guaranteed to converge to an optimal solution, i.e., for all $\epsilon \in \mathbb{R}^+$, there exists $k_\epsilon > 0$ such that, for all $k \geq k_\epsilon$, $f_0(x^{(k)}) - p^* \leq \epsilon$ holds, that is, the $k_\epsilon$-th iterate provides an $\epsilon$-suboptimal solution of (5).

The convergence result of Proposition 4.5 is stated under the assumption of infinite precision computation. For finite precision computation, such as floating point computation, $\epsilon$ cannot be made arbitrarily small. For example, $\epsilon$ must be larger than the machine precision $\epsilon_{\text{mach}}$ [45], [46], which is typically of order $10^{-16}$ in double precision computer arithmetic. In general, results from roundoff-error analysis of iterative numerical methods [45], [46] guarantee that convergence is still achieved in the sense of Proposition 4.5 for all $\epsilon \geq \epsilon_{\min}$, where $\epsilon_{\min} > 0$ is a linear function of $\epsilon_{\text{mach}}$ that (weakly) depend on the problem instance. Similarly, $\delta$ in Definition 4.4 cannot be made arbitrarily small; in what fallows, we assume that $\delta$ is bounded below by an appropriate constant $\delta_{\min} > 0$.

Proposition 4.5 provides sufficient conditions for the convergence of a numerical optimization method to an $\epsilon$-suboptimal solution. A projected gradient method may not be efficient, in general, for constrained convex problems such as (5). However, similar guarantees can also be obtained for other methods, such as the barrier method [14], which, together with the broader family of interior point methods, is the cornerstone of state-of-the-art convex optimization routines. For example, it is possible to provide theoretical bounds to the number of steps needed by Newton's method to solve a convex optimization problem as well as the number of steps required to reach a given precision [14], [45].

## C. Properties of Monotone SMC Formulas

Monotone SMC formulas have the desirable property that the corresponding satisfiability problem 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 4.6 (Monotone Convex Expansion):** Let $\varphi$ be an SMC formula, $C$ be the set of convex constraints, appearing in $\varphi$, and $|C|$ its cardinality. We define the propositional abstraction of $\varphi$ to be the formula $\varphi_B$ obtained from $\varphi$ by replacing each convex constraint with a Boolean variable $a_i$, $i \in \{1, \ldots, |C|\}$. We further define the monotone convex expansion of $\varphi$ to be the formula $\varphi'$ defined as:

$$\varphi' = \varphi_B \wedge \bigwedge_{i=1}^{|C|} (a_i \rightarrow (g_i(x) \lhd 0)), \tag{6}$$

where $(g_i(x) \lhd 0)$ denotes a convex constraint, appearing in $\varphi$, as defined in Section IV-A.

**Proposition 4.7:** Let $\varphi'$ be the monotone convex expansion of a monotone SMC formula $\varphi$, defined as in (6), where $\varphi_B$ is the propositional abstraction of $\varphi$. Then, the following properties hold:

1) $\varphi$ and $\varphi'$ are equisatisfiable, i.e., if $(b^*, x^*, a^*)$ is a model (a satisfying assignment) for $\varphi'$, then $(b^*, x^*)$ is a model for $\varphi$; if $\varphi'$ is unsatisfiable, then so is $\varphi$;

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



2) any Boolean assignment for $\varphi_B$ turns $\varphi'$ into a conjunction of convex constraints;
3) the satisfiability problem for $\varphi'$, hence $\varphi$, can always be cast as the feasibility problem for a finite disjunction of convex programs.

Proposition 4.7 directly follows from the monotonicity of $\varphi$. In preparation for the proof, we introduce the following definition and lemma.

**Definition 4.8 (Monotone Formula):** A propositional formula $\varphi_B$ is monotone in its literal $l$ if the literal $l$ is always positive (i.e., without negations).

For a monotone formula it is straightforward to show the following property.

**Lemma 4.9:** Let $\varphi_B$ be a propositional formula monotone in the literal $l$ and $\nu$ be a model for $\varphi_B$. If $\nu_l$ is the assignment obtained by $\nu$ by asserting $l$ and by keeping unaltered the truth value assigned to the other propositions, then $\nu_l$ is also a model for $\varphi_B$, i.e., $\nu_l \models \varphi_B$.

We are now ready to prove Proposition 4.7.

**Proof (Proposition 4.7):** 1) Formula $\varphi$ is equisatisfiable to a formula $\varphi''$ constructed as follows:

$$\varphi'' = \varphi_B \wedge \bigwedge_{i=1}^{|C|} (a_i \rightarrow g_i(x) \lhd 0) \wedge (\neg a_i \rightarrow g_i(x) \not\lhd 0)\,. \qquad (7)$$

Moreover, because $a_i$ is true if and only if $(g_i(x) \lhd 0)$ is satisfied in (7), there is a one-to-one correspondence between the models of $\varphi$ and the ones of $\varphi''$. Since $\varphi''$ includes all the clauses in $\varphi'$, we conclude that, if $\varphi'$ is unsatisfiable, then $\varphi''$, hence $\varphi$, is also unsatisfiable.

On the other hand, let $(b^*, x^*, a^*)$ be a model for $\varphi'$. Since $\varphi$ is monotone SMC, $\varphi_B$ is monotone in the $a_i$ (Definition 4.8). Then, it is always possible to construct a new model $(b^*, x^*, a^{**})$ for $\varphi''$, such that $a_i^{**}$ is true if $(g_i(x^*) \lhd 0)$ is satisfied, and false otherwise. In fact, such a procedure can only increase the number of variables that are asserted in $a^*$, which does not impact the satisfiability of $\varphi_B$, because of its monotonicity property (Lemma 4.9), while making all the implication clauses in both $\varphi'$ and $\varphi''$ true. The assignment $(b^*, x^*, a^{**})$ is, therefore, a model for $\varphi''$. Finally, by projecting out $a^{**}$, we obtain that $(b^*, x^*)$ is also a model for $\varphi$. Formulas $\varphi$ and $\varphi'$ are then equisatisfiable.

2) A satisfying assignment for $\varphi_B$ either trivially satisfies an implication clause $i$ on the right side of (6) ($a_i$ evaluates to false) or turns it into the convex constraint $(g_i(x) \lhd 0)$ ($a_i$ evaluates to true). Then, $\varphi'$ is satisfiable if and only if the conjunction of convex constraints such that the associated variable $a_i$ is asserted is satisfied.

3) By property 1) and 2), since the satisfying assignments of $\varphi_B$ are finite, a model for $\varphi'$, hence $\varphi$, can always be found by solving a finite set of convex programs, which is in the worst case equal to the number of satisfying assignments for $\varphi_B$. Remarkably, if no variables in $a$ need to be asserted for $\varphi_B$ to be satisfiable, then the satisfiability of $\varphi'$ does not depend on its real variables, i.e., $\varphi'$ is satisfiable for all $x \in \mathcal{D} \subseteq \mathbb{R}^n$.

By Proposition 4.7, any monotone SMC formula $\varphi$ can be solved by casting and solving a disjunction of convex programs. We will use this property to construct our decision procedure in Section V. 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 4.10:** Let $\varphi$ 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, $\varphi$ can be posed as a monotone SMC formula.

**Proof (Theorem 4.10):** We can always encode $\varphi$ as a finite disjunction of convex programs as follows:

$$\left(\bigwedge_{i=1}^{p_1} (g_i^{(1)}(x) \lhd 0) \wedge b^{(1)}\right) \vee \left(\bigwedge_{i=1}^{p_2} (g_i^{(2)}(x) \lhd 0) \wedge b^{(2)}\right) \vee \ldots \vee \left(\bigwedge_{i=1}^{p_r} (g_i^{(r)}(x) \lhd 0) \wedge b^{(r)}\right) \qquad (8)$$

where we assume that $r$ convex feasibility problems are associated with distinct assignments over the Boolean variables $b_1, \ldots, b_r$ such that $b^{(j)} = \bigwedge_{i=1}^{j-1} \neg b_i \wedge b_j \wedge \bigwedge_{i=j+1}^{r} \neg b_i$. Each convex problem $\mathcal{P}_j$ is a conjunction of $p_j$ convex constraints, each of the form $(g_i^{(j)}(x) \lhd 0)$, with $i \in \{1, \ldots, p_j\}$. By the distributive property of disjunction with respect to conjunction, we can then translate the disjunction of terms into a conjunction of clauses as in the syntax in (2). Therefore, (8) encodes a monotone SMC formula. □

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

**Corollary 4.11:** Monotone SMC formulas include any Boolean Satisfiability (SAT) problem instance and any Mixed Integer Convex (MIC) feasibility problem instance as a particular case.

**Proof (Corollary 4.11):** An SAT problem is trivially a particular case of an SMC problem. To prove that this is also the case for an MIC problem, we observe that an MIC feasibility problem can be modeled as the feasibility of a conjunction of convex constraints in $(b, x)$ as follows:

$$\bigwedge_{i=1}^{p} (g_i(b, x) \lhd 0)$$

where $p$ is the number of constraints. By enumeration of all possible assignments $b^{(1)}, \ldots, b^{(r)}$ to the Boolean

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



variables, we can always expand the above MIC into a finite disjunction of convex programs as follows:

$$\left(\bigwedge_{i=1}^{p}(g_i(b^{(1)},x)\lhd 0)\wedge d^{(1)}\right)\vee\left(\bigwedge_{i=1}^{p}(g_i(b^{(2)},x)\lhd 0)\wedge d^{(2)}\right)\\ \ldots\vee\left(\bigwedge_{i=1}^{p}(g_i(b^{(r)},x)\lhd 0)\wedge d^{(r)}\right) \tag{9}$$

where, for all $i\in\{1,\ldots,p\}$ and $j\in\{1,\ldots,r\}$, $g_i(b^{(j)},x)$ is convex by definition, and the conjunction of atoms $d^{(j)}=\wedge_{i=1}^{m}b_i^{(j)}$ evaluates to true if and only if $b=b^{(j)}$. Therefore, by Theorem 4.10, the satisfiability of (9), hence the feasibility of the original MIC problem, can be captured as the satisfiability of a monotone SMC formula following the syntax in (2).

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 [16]. For instance, $\varphi$ 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 $\varphi$, but violates $c$.

### D. Examples

The design problems in Section III can all be encoded as satisfiability problems for monotone SMC formulas according to Definition 4.2, for which the properties in Section IV-C hold. We provide details for these encodings as follows.

*1) ARPOD Mission:* Let $L$ be the time horizon and $b_k^i$ a binary variable that evaluates to 1 (true) if and only if the system is in mode $i$ at time $k$. We can require the system to be in one and only one mode at each time with the following pseudo-Boolean constraint:

$$\sum_{i=1}^{3}b_k^i=1\quad\forall\,k\in\{0,\ldots,L\}. \tag{10}$$

Moreover, if the system is in mode $i$ at time $k$, it can only stay in mode $i$ or transition to a direct successor mode at time $k+1$, i.e.,

$$b_0^1\wedge b_L^3\wedge\bigwedge_{k=0}^{L-1}(b_k^1\rightarrow b_{k+1}^1\vee b_{k+1}^2)\wedge(b_k^3\rightarrow b_{k+1}^3). \tag{11}$$

The rendezvous and docking phases are analyzed in a relative coordinate frame, known as the Hill's frame, describing the difference in position and velocity between the chaser and target spacecraft. Let $\zeta_1$, $\zeta_2$, and $\zeta_3$ be the coordinates of the chaser spacecraft in the Hill's frame; then, in mode 1 and 2, the translational motion of the chaser must satisfy the Clohessy-Wiltshire-Hill (CWH) equations, which can be expressed in linear, time-invariant, discrete time state-space form for a given sampling time. The docked spacecraft pair also obeys a similar set of equations in Phase 3, where the dynamics change to account for the increased mass of the pair. Overall, the set of constraints associated with the dynamics can be formulated as follows:

$$b_k^i\rightarrow x_{k+1}=A_ix_k+B_iu_k=\text{CWH}(x,u,\gamma,m_i,t_s^i) \tag{12}$$

$\forall\,k\in\{0,\ldots,L-1\}$, $\forall\,i\in\{1,2,3\}$, where $m_i$ is the spacecraft mass in mode $i$, $\gamma=\sqrt{\nu/\beta^3}$ is the mean-motion of the target, $\nu$ is the Earth's gravitational constant, $\beta$ is the length of the semi-major axis of the target's orbit, and $t_s^i$ is the sampling time in mode $i$. For a two-degrees-of-freedom (2DOF) case, we define $x=(\zeta_1,\zeta_2,\dot{\zeta}_1,\dot{\zeta}_2)^T$ and $u=(F_{\zeta_1},F_{\zeta_2})^T$, $F_{\zeta_1}$ and $F_{\zeta_2}$ being the thrust forces applied to the chaser spacecraft. Analogous definitions hold for the 3DOF case. The state vector $x$ is made up of both positions and velocities along the $\zeta_1$ and $\zeta_2$ axes. Moreover, for all phases, the constraint on the maximum control input at any single point in time is

$$\|u_k\|_\infty\leq\bar{u}\quad\forall\,k\in\{0,\ldots,L-1\} \tag{13}$$

where $\bar{u}$ is the upper limit on the thrust that can be produced in each of the axial directions.

The relative displacement vector from the target to the chaser is defined in the Hill's frame as $\zeta=(\zeta_1,\zeta_2)^T$ in 2DOF. The magnitude of this displacement vector in the $\ell_\infty$-norm is $\rho=\|\zeta\|_\infty$. The position of the chaser satellite at the initial time is $\rho_0$, within 10 km of the target spacecraft. The chaser begins its maneuvers in Phase 1 while $\rho_0\geq\rho\geq\rho_d$, $\rho_d=100$ m is the separation distance at which docking starts. This condition translates into the additional constraints:

$$b_k^1\rightarrow(\rho_k\leq\rho_0)\wedge(\rho_d\leq\rho_k)\quad\forall\,k\in\{0,\ldots,L\}. \tag{14}$$

After the chaser reaches a distance $\rho<\rho_d$, the docking phase (Phase 2) is initiated, i.e.,

$$b_k^2\rightarrow\rho_d>\rho_k\quad\forall\,k\in\{0,\ldots,L\}. \tag{15}$$

In Phase 2, the chaser attempts to reduce $\rho$ to zero while remaining in the LOS region, $X_{LOS}$, and maintaining a slow velocity so as to reduce impact forces upon docking. We capture the LOS constraint as a second order cone (convex) constraint as follows:

$$b_k^2\rightarrow\|(\zeta_{1,k},\zeta_{2,k})\|_2\leq\frac{c^T(\zeta_{1,k},\zeta_{2,k})^T}{\|c\|_2\cos\left(\frac{\theta}{2}\right)} \tag{16}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



$\forall\, k \in \{0, \ldots, L\}$, where $c$ and $\theta$ are, respectively, the cone axis and aperture, and $(\zeta_{1,k}, \zeta_{2,k})^T$ is the displacement vector at time $k$. Additionally, the chaser's velocity must be kept under the specified value, which can be captured by a convex $\ell_2$-norm constraint of the form

$$b_k^2 \rightarrow \|(\dot{\zeta}_{1,k}, \dot{\zeta}_{2,k})\|_2 \leq \bar{V} \quad \forall\, k \in \{0, \ldots, L\} \tag{17}$$

in the 2DOF version. Finally, the chaser must dock to the target at the end of Phase 2, i.e.,

$$b_k^2 \wedge b_{k+1}^3 \rightarrow x_k = x_{\text{docked}} \quad \forall\, k \in \{0, \ldots, L-1\}. \tag{18}$$

Once the chaser spacecraft docks (i.e., $\rho = 0$ m), both spacecraft enter Phase 3, where the joint assembly must move to the relocation position. The end location must be the relocation spot, i.e.,

$$x_L = x_{\text{relocation}}. \tag{19}$$

The conjunction of constraints (10)–(19) generates a formula $\varphi_{\text{ARPOD}}$ that can be captured by the grammar in Definition 4.2. Formula $\varphi_{\text{ARPOD}}$ is then monotone SMC.

*2) Motion Planning:* For simplicity, we present an encoding for the basic reach-avoid problem under the assumptions in Section III-B. However, our approach extends to motion planning from generic LTL specifications [36] by following the bounded model checking encoding techniques [47] to formulate the discrete planning problem.

We assume that the workspace regions are described by polyhedra, as shown in Fig. 5 (left), and captured by affine 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 $L$, let $b_k^i$ be a Boolean variable that is asserted if and only if the robot is in region $i$ at time $k$. We can then encode the constraints for the controller using the following logic formula $\varphi_{\text{MP}}$:

$$\begin{aligned}
\varphi_{\text{MP}} :\, &= b_0^{\text{start}} \text{ (initial partition)} \\
&\wedge\, b_L^{\text{goal}} \text{ (goal partition)} \\
&\wedge \left( b_k^i \rightarrow \bigvee_{i' \in \Pi(i)} b_{k+1}^{i'} \right) \ \forall\, k \in \{0, \ldots, L-1\}, \\
&\qquad\qquad i \in \{1, \ldots, m\} \\
&\text{(transition relation)} \\
&\wedge \left( \sum_{i=1}^{m} b_k^i = 1 \right) \ \forall\, k \in \{0, \ldots, L\} \\
&\text{(mutual exclusion)} \\
&\wedge\, (x_{k+1} = Ax_k + Bu_k) \ \forall\, k \in \{0, \ldots, L-1\} \\
&\text{(robot dynamics)} \\
&\wedge\, (\|u_k\| \leq \overline{u}) \ \forall\, k \in \{0, \ldots, L-1\} \\
&\text{(control bounds)} \\
&\wedge\, (x_0 = \overline{x}) \text{ (initial state)} \\
&\wedge \left( b_k^i \rightarrow P_i x_k + q_i \leq 0 \right) \ \forall\, k \in \{0, \ldots, L\}, \\
&\qquad\qquad i \in \{1, \ldots, m\} \\
&\text{(region constraints)}
\end{aligned}$$

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 $\bar{u}$ is the maximum feasible magnitude $\|u_k\|$ (e.g., $\ell_2$- or $\ell_\infty$-norm) of the control input at time $k$. We observe that $\varphi_{\text{MP}}$ is a monotone SMC formula by Definition 4.2.

We further observe that the satisfying assignments of the Boolean abstraction of $\varphi_{\text{MP}}$ are characterized by an ordering imposed by the feasible runs of the transition system in Fig. 5. If a sequence of regions $\sigma$ is feasible, then so is any prefix sequence of $\sigma$. We will call the formulas encoding such a scenario *POM formulas* and provide the formal definition in Section VI-C. POM formulas 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. Scalable decision procedures for monotone SMC formulas can be developed based on efficient methods for detecting minimal sets of conflicting convex constraints. For POM formulas, this task reduces to solving only one convex program.

Our formulation differs from classical approaches to reach-avoid problems [48]–[50], e.g., based on the solution of a Hamilton–Jacobi–Isaacs equation or the computation or approximation of reachable sets (see, e.g., [4] for a survey of methods and tools). Rather than formulating a complete, general optimization problem, which may be computationally challenging, we focus on solving a special case accurately and efficiently. We then aim at leveraging this result as a building block to solve more general problems, e.g., by supporting complex LTL specifications, through abstraction and refinement techniques.

Finally, similar encoding techniques can be used for a *multirobot motion planning* problem. In this scenario, constraints such as the ones in $\varphi_{\text{MP}}$ must be generated and conjoined for each robot. Moreover, additional constraints are needed to ensure collision avoidance. By assuming a 3-D workspace and $N$ robots, for each pair of robots at each time we can create pairs of Boolean variables $\{(f_{kd}^{pq}, g_{kd}^{pq}) | k \in \{0, \ldots, L\}, d \in \{1, 2, 3\}, p, q \in \{1, \ldots, N\}, p \neq q\}$ and then encode the collision

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



avoidance conditions via the conjunction of the following constraints:

$$\forall\, p, q \in \{1, \dots, N\}, p \neq q, \forall\, k \in \{0, \dots, L\}:$$

$$f^{pq}_{kd} \rightarrow h^d_{\mathcal{X}\rightarrow\mathcal{W}}(x^p_k) - h^d_{\mathcal{X}\rightarrow\mathcal{W}}(x^q_k) \geq \epsilon \quad \forall d \in \{1,2,3\} \tag{20}$$

$$g^{pq}_{kd} \rightarrow -h^d_{\mathcal{X}\rightarrow\mathcal{W}}(x^p_k) + h^d_{\mathcal{X}\rightarrow\mathcal{W}}(x^q_k) \geq \epsilon \quad \forall d \in \{1,2,3\} \tag{21}$$

$$\sum_{d=1}^{3} (f^{pq}_{kd} + g^{pq}_{kd}) \geq 1 \tag{22}$$

where $h^d_{\mathcal{X}\rightarrow\mathcal{W}}(.)$ is the natural projection of the state space onto the $d$-th dimension of the workspace. The implications in (20) and (21) require the displacement between robot $p$ and robot $q$ to be larger than or equal to $\epsilon \in \mathbb{R}^+$ in any of the directions along the $d$-th axis. Constraint (22) requires that, across the three dimensions, at least one of the constraints in (20) and (21) be active. Constraints (20)–(22) are, again, all captured by the grammar for monotone SMC formulas in Definition 4.2.

*3) Secure State Estimation:* We finally show that the secure state estimation problem under the assumptions of Section III-C can also be encoded as the satisfiability problem for a monotone SMC formula. Let $Y_1, Y_2, \dots, Y_p$ be the set of $p$ sensor measurements taken over a time window out of a linear dynamical system. These measurements are then a function of the system state, where

$$Y_i = \begin{cases} H_i x, & \text{if sensor } i \text{ is attack-free} \\ H_i x + \alpha_i, & \text{if sensor } i \text{ is under attack} \end{cases} \tag{23}$$

and $\alpha_i$ models the attack injection.

We are interested in reconstructing the state of the dynamical system even if up to $k$ sensors are maliciously corrupted. We can encode this problem by introducing binary indicator variables $b_i$ that evaluate to 1 if and only if the sensor $i$ is attacked. We therefore obtain the following monotone SMC formula:

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

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 $\upsilon \in \mathbb{R}^+$.

## V. 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 [15]. The SAT solver efficiently reasons about combinations of Boolean and pseudo-Boolean constraints, using the DPLL algorithm [22], 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 $\varphi'$ of a monotone formula $\varphi$ translates into a conjunction of convex constraints for any Boolean assignments by Proposition 4.7, we can generate queries to a theory solver that are always in the form of conjunctions of convex constraints and can be efficiently solved by convex programming. We can then adopt a lazy SMT approach, by breaking our decision task into two simpler tasks, respectively, over the Boolean and convex domains.

**Algorithm 1:** SMC **Input:** $\varphi, \delta$
**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;
13:    end if
14:  end if
15: end while
```

As illustrated in Algorithm 1, we start by generating the propositional abstraction $\varphi_B(b, a)$ of $\varphi$. We denote by $\mathcal{M}$ the map that associates each convex constraint in $\varphi$ with an auxiliary variable $a_i$. By only relying on the Boolean structure of $\varphi_B$, SAT-SOLVE may either return UNSAT or propose a satisfying assignment $\mu$ 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 $\varphi_B$; we denote by $\text{supp}(a^*)$ the set of indices of auxiliary variables $a_i$ which are asserted in $a^*$. This Boolean assignment is then used by $\mathcal{C}$-SOLVE to determine whether there exist 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:

$$\text{find } x \text{ s.t. } g_i(x) \lhd 0 \quad \forall\, i \in \text{supp}(a^*) \tag{24}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



which is the 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:

$$\min_{\substack{s_1,\dots,s_L\in\mathbb{R}\\ x\in\mathbb{R}^n}} \quad \sum_{i=1}^{L} |s_i| \text{ s.t. } \quad g_{j_i}(x) \lhd s_i, \quad i=1,\dots,L \tag{25}$$

where $L$ is the cardinality of $\text{supp}(a^*)$ and $j_i$ spans $\text{supp}(a^*)$ as $i$ varies in $\{1,\dots,L\}$.

Problem (25) is equivalent to (24), as it tries to minimize the infeasibilities of the constraints by pushing each slack variable to be as much as possible close to zero. Problem (25) is also a convex program whose optimum value is achieved and it is zero if and only if the original set of constraints in (24) is feasible. Therefore, if the optimal cost is zero (in practice, the condition $\sum_{i=1}^{L}|s_i| \leq \delta$ is satisfied for a "small" $\delta \in \mathbb{Q}^+$), then $\mu$ is indeed a valid assignment, an optimal ($\delta$-suboptimal) solution $x^*$ is found, and our algorithm terminates with SAT and provides the solution $(x^*, b)$, denoted by $\eta(b,x)$ in Algorithm 1. Otherwise, an UNSAT certificate $\varphi_{ce}$ is generated in terms of a new Boolean clause explaining which auxiliary variables should be negated since the associated convex constraints are conflicting. The trivial certificate

$$\varphi_{\text{trivial-ce}} = \bigvee_{i\in\text{supp}(a^*)} \neg a_i \tag{26}$$

can always be provided, encoding the fact that at least one of the auxiliary variables indexed by an element in $\text{supp}(a^*)$ should actually be negated. The augmented Boolean problem consisting of the original formula $\varphi_B$ conjoined with the generated certificate $\varphi_{ce}$ 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 proposition summarizes the formal guarantees of Algorithm 1 with the trivial certificate (26).

**Proposition 5.1:** Let $\varphi$ be a monotone SMC formula and $\delta \in \mathbb{Q}^+$ a user-defined tolerance used by $\mathcal{C}$-SOLVE.CHECK in Algorithm 1 to accommodate numerical errors. Algorithm 1 with the UNSAT certificate $\varphi_{ce}$ in (26) is $\delta$-complete.

**Proof (Proposition 5.1):** Since $\varphi$ and its monotone convex expansion $\varphi'$ are equisatisfiable, we can directly apply Algorithm 1 to the satisfiability problem for $\varphi'$. In the worst case, Algorithm 1 executes a number of iterations equal to the number of satisfying assignments of $\varphi_B$. Moreover, by Theorem 4.7, at each iteration, $\mathcal{C}$-SOLVE.CHECK is guaranteed to solve a convex program. At each iteration, either $\mathcal{C}$-SOLVE.CHECK terminates with a feasible solution or the current Boolean variable assignment is excluded from the set of satisfying assignments for $\varphi_B$, which guarantees progress. Since the number of the assignments for $\varphi_B$ is finite, Algorithm 1 will always terminate.

We observe that (25) does not directly satisfy all the assumptions of Proposition 4.5, but can be easily reformulated, using a standard change of variables [14], so that the objective function is continuously differentiable over the closed convex set generated by the constraints, where the $g_{j_i}$, $i = 1,\dots,L$, are all closed convex functions. Problem (25) is also feasible, since, for any $x$, we can always select values for $s_1,\dots,s_L$ such that the constraints are satisfied. Moreover, the optimal value is achieved. By Proposition 4.5, it is then possible to provide a $\delta$-suboptimal solution of (25).

If (24) is feasible, meaning that the sum of the absolute values of the slack variables in the SSF problem at optimum is bounded by $\delta$, then we terminate with SAT. If (24) is infeasible and the optimum $p^*$ for the SSF problem is larger than $\delta$, then SAT-SOLVE.CHECK correctly terminates with UNSAT. On the other hand, if (24) is infeasible and $0 < p^* \leq \delta$ holds, then SAT-SOLVE.CHECK can either terminate with UNSAT or SAT. In all cases, SAT-SOLVE.CHECK terminates correctly in the sense of Definition 4.4. Finally, based on the guarantees of SAT-SOLVE.CHECK, Algorithm 1 terminates correctly with $\delta$-SAT or UNSAT, hence it is $\delta$-complete. □

The worst case bound on the number of iterations in Algorithm 1 is exponential in the number of convex constraints $|C|$. 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.

## VI. GENERATING SMALL CERTIFICATES

### A. 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) [51] of constraints, defined as follows.

**Definition 6.1 (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 and 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

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



variables in $\varphi_B$ that are associated with a convex constraint in an IIS $I$. Then, once $I$ is found, a minimal certificate can be generated as

$$\varphi_{\text{IIS-ce}} = \bigvee_{i \in \mathcal{I}} \neg a_i. \tag{27}$$

Many techniques proposed in the literature to isolate IISs are based on either adding constraints, one by one or in groups, to a feasible set of constraints, until an inconsistency is detected (*additive method*), or by deleting constraints from the original problem, until the constraint set becomes feasible (*deletion filtering*) [51], [52]. Usually, a combination of two or more filtering methods can guarantee that all of the constraints in an inconsistent set are essential, hence the set is minimal, and none can be excluded from the set. An IIS with the smallest cardinality indeed guarantees that the length of the certificate is minimum, which can dramatically reduce the search space in Algorithm 1. However, isolating a minimum IIS can be very expensive [52], [53]. In the worst case, as shown in Table 1, finding a minimum cardinality IIS can require solving a feasibility problem for each subset of constraints in $S$, which is exponential in the size $|S|$.

If, instead, we are interested in only one IIS, rather than the smallest one, then the problem can be solved by searching over the constraints and solving a number of feasibility problems that is linear in $|S|$. This search algorithm can be accelerated significantly by divide-and-conquer strategies that successively decompose the overall problem and filter the constraints in groups [52]. Depending on the selected decomposition, it is then possible to isolate an IIS by solving a number of feasibility problems that grows with the logarithm of $|S|$ and is linear in the cardinality $|I|$ of the IIS. These techniques provide the technological basis for conflict explanations in industrial constraint programming tools [52] and can support an interactive approach to isolate an IIS based on user preferences between constraints. Overall, the following proposition summarizes the correctness guarantees of Algorithm 1 with an IIS-based certificate (27).

**Proposition 6.2:** Let $\varphi$ be a monotone SMC formula and $\delta \in \mathbb{Q}^+$ a user-defined tolerance used by $\mathcal{C}$-SOLVE.CHECK and $\mathcal{C}$-SOLVE.CERT in Algorithm 1 to accommodate numerical errors. Algorithm 1 with the UNSAT certificate in (27) is $\delta$-complete.

**Table 1** Algorithms for Certificate Generation: Number of Convex Programs Needed to Generate a Certificate and Length of Generated Certificate. $|S|$ Is Number of Constraints in Convex Program

| Certificate | # Convex Programs | Length $\ell$ |
|---|---|---|
| Trivial | 1 | $\lvert S\rvert$ |
| Minimum IIS-Based | Exponential in $\lvert S\rvert$ | $\lvert I\rvert_{\min}$ |
| IIS-Based | Logarithmic in $\lvert S\rvert$<br>Linear in $\lvert I\rvert$ | $\lvert I\rvert_{\min} \leq \lvert I\rvert \leq \lvert S\rvert$<br>(minimal) |
| SSF-Based | Logarithmic in$\lvert S\rvert$<br>Linear in $\ell$ | $\lvert I\rvert_{\min} \leq \ell \leq \lvert S\rvert$ |
| Prefix-Based | 1 | $\lvert I\rvert_{\min} \leq \ell \leq \lvert S\rvert$ |

**Proof (Proposition 6.2):** The proof proceeds along the lines of the one for Proposition 5.1. □

In the following, we describe an algorithm that rather approximates an IIS, i.e., it can generate a small, albeit nonminimal, set of conflicting constraints by solving a number of convex programs that is usually smaller than the one needed for IIS-based certificates.

## B. SSF-Based Certificates

A computationally efficient alternative to IIS-based certificates is to directly exploit the information in the slacks of the SSF problem (25) to rank the constraints and guide the search for smaller conflicting sets. Leveraging information from the slack variables is also part of the *elastic* and *sensitivity filtering* methods, which have been proposed for explaining conflicts in linear programs [51] and only recently extended to nonlinear programs [30].

If a constraint $k$ is associated with a nonzero optimal slack, $|s_k^*| > 0$, then it is a member of one of the IIS in problem (24). However, the set of all the constraints with a nonzero 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, even if the returned conflict set may not be minimal. 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 $\mathcal{I}_\text{temp}$ 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 $\mathcal{I}_\text{temp}$ 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 $\mathcal{I}_\text{temp}$.

The procedure summarized in Algorithm 2 solves a number of convex programs that is linear in the number of constraints $|S|$ in (24). In fact, the cardinality of the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



**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**
    $\varphi_{\text{SSF-ce}} := \bigvee_{i \in \mathcal{I}_temp} \neg a_i$;
  **else**
    counter := counter + 1;
    $\mathcal{I}_temp := \mathcal{I}_temp \cup \mathcal{I}_max_{counter}$;
  **end if**
**end while**
**return** $\varphi_{\text{SSF-ce}}$

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. However, in practice, the time needed to generate an SSF-based certificate is smaller than the time required for an IIS-based certificate, since we only need to construct an approximation of an IIS. Moreover, Algorithm 2 can also benefit from divide-and-conquer approaches [52] that can partition the constraints in different ways to accelerate the search, and lower the number of feasibility checks to grow as the logarithm of $|S|$. The following proposition summarizes the correctness guarantees of Algorithm 1 with the SSF-based certificate in Algorithm 2.

**Proposition 6.3:** Let $\varphi$ be a monotone SMC formula and $\delta \in \mathbb{Q}^+$ a user-defined tolerance used by $\mathcal{C}$-SOLVE.CHECK and $\mathcal{C}$-SOLVE.CERT-SSF in Algorithm 1 to accommodate numerical errors. Algorithm 1 with the UNSAT certificate in Algorithm 2 is $\delta$-complete.

**Proof (Proposition 6.3):** The proof proceeds along the lines of the one for Proposition 5.1. □

## C. Prefix-Based Certificate

Under additional monotonicity assumptions on the structure of $\varphi$ 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 *prefix-ordered formula* below. For convenience, in what follows, we use the notation $b = 1$, $b = 0$, and $b = *$ to indicate that $b$ is asserted, negated, or unassigned, respectively. We then recall the definition of the *restriction* of a Boolean formula.

**Definition 6.4 (Restriction):** We call a function $\rho : \{1, 2, \ldots, m\} \to \{0, 1, *\}$ a restriction. Given a Boolean formula $\varphi(b_1, b_2, \ldots, b_m)$, we call $\varphi$ restricted by $\rho$, written $\varphi \restriction \rho$, the formula obtained after assigning to each $b_i$ of $\varphi$ the $i$-th value (character) of $\rho$. Given a satisfying assignment $\mu$, we also call restriction induced by $\mu$ the restriction $\rho_\mu$ that assigns its $i$-th value to 0 if $b_i$ is negated by $\mu$, and to $*$ otherwise.

For example, given $\varphi(b_0, b_1, b_2, b_3, b_4) := (b_0 \vee b_1) \wedge (b_1 \to (b_2 \vee b_3)) \wedge b_4$ and a restriction $\rho$ such that $\rho(0) = \rho(3) = 0$, $\rho(4) = 1$ and $\rho(1) = \rho(2) = *$, we have $\varphi \restriction \rho = \varphi(0, b_1, b_2, 0, 1) := b_1 \wedge (b_1 \to b_2)$. Given a satisfying assignment $\mu$ such that $[\![b_0]\!]_\mu = [\![b_3]\!]_\mu = \bot$, $[\![b_1]\!]_\mu = [\![b_2]\!]_\mu = [\![b_4]\!]_\mu = \top$, we also have $\varphi \restriction \rho_\mu := b_1 \wedge (b_1 \to b_2) \wedge b_4$.

**Definition 6.5 (Prefix-Ordered Formula):** A Boolean formula $\varphi(b_1, b_2, \ldots, b_m)$ is said to be prefix-ordered with respect to $(b_1, \ldots, b_m)$, with $m \geq 2$, if we have

$$\varphi \to b_1 \wedge (b_1 \to b_2) \wedge (b_2 \to b_3) \wedge \ldots \wedge (b_{m-1} \to b_m). \quad (28)$$

By Definition 6.5, a prefix-ordered formula entails a chain of implications. While its satisfying assignment, $(1, \ldots, 1)$, can be trivially determined, we are rather interested in the structure of the *falsifying assignments*, since they are relevant to the construction of UNSAT certificates. A prefix-ordered formula has a set of falsifying assignments that can be ordered based on the number of consecutive asserted variables in their prefixes before the occurrence of a negated variable. In other words, we call *implicants* of $\neg\varphi$ the formulas $\varphi_{ce}$ such that $\varphi_{ce} \to \neg\varphi$. Such implicants can be interpreted as explanations for the infeasibility of $\varphi$. Definition 6.5 states that the implicants of $\neg\varphi$ includes terms of the following forms: $\neg b_1$, $b_1 \wedge \neg b_2$, $b_1 \wedge b_2 \wedge \neg b_3, \ldots, b_1 \wedge b_2 \wedge \ldots \wedge b_{m-1} \wedge \neg b_m$. Moreover, each of these implicants is a prime implicant, as its number of literals cannot be further reduced. We now extend this notion of prefix-based ordering to a subclass of convex formulas of interest to us, which we term prefix-ordered monotone SMC (POM) formulas.

**Definition 6.6 (Prefix-Ordered Monotone SMC Formula):** Let $\varphi_B(b, a)$ be the propositional abstraction of a monotone SMC formula $\varphi$. We say that $\varphi$ is prefix-ordered monotone SMC (POM) with respect to ($\mu$,$\lambda$,$\kappa$) if there exists a satisfying assignment $\mu$, an associated restriction $\rho_\mu$, an ordering (renaming) $\lambda : \mathcal{J} \to \mathcal{J}$ over the index set $\mathcal{J}$ of the binary variables $b$, an ordering $\kappa : \mathcal{I} \to \mathcal{I}$ over the index set $\mathcal{I} = \{1, \ldots, |C|\}$ of the auxiliary variables $a$, and $m \geq 2$ such that:

i) $\varphi_B \restriction \rho_\mu \to \psi(b_{\lambda(1)}, \ldots, b_{\lambda(m)}) \bigwedge_{i=1}^{m} (b_{\lambda(i)} \to a_{\kappa(i)})$;
ii) $\psi$ is prefix-ordered with respect to $(b_{\lambda(1)}, \ldots, b_{\lambda(m)})$.

**Example 1 (POM Formulas):** Consider an SMC formula $\varphi_1$ and its propositional abstraction defined as follows:

$$\varphi_{B1} : = (b_0 \vee b_1) \wedge (b_0 \to b_3) \wedge (b_1 \to (b_2 \vee b_3)) \wedge (b_2 \to b_4) \wedge (b_3 \to b_4) \wedge \bigwedge_{i=0}^{4} (b_i \to a_i)\,.$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



Such a formula is analogous to those obtained from the encodings of motion planning problems discussed in Section IV-D2. Consider the restriction $\rho_1$ associated with the satisfying assignment $\mu_1$ such that $[\![b_0]\!]_{\mu_1} = [\![b_2]\!]_{\mu_1} = \bot$ and $[\![b_1]\!]_{\mu_1} = [\![b_3]\!]_{\mu_1} = [\![b_4]\!]_{\mu_1} = \top$, i.e., $\rho_1(0) = \rho_1(2) = 0$ and $\rho_1(1) = \rho_1(3) = \rho_1(4) = *$. We obtain

$$\varphi_{B1} \restriction \rho_1 : = b_1 \wedge (b_1 \rightarrow b_3) \wedge (b_3 \rightarrow b_4) \wedge (b_1 \rightarrow a_1) \\ \wedge (b_3 \rightarrow a_3) \wedge (b_4 \rightarrow a_4)\,.$$

By Definition 6.6, we conclude that $\varphi_1$ is POM with respect to $\mu_1$, the Boolean variables $(b_1, b_3, b_4)$, and the associated auxiliary variables $(a_1, a_3, a_4)$.

Consider now an SMC formula $\varphi_2$ whose propositional abstraction is defined as follows:

$$\varphi_{B2} := (b_0 \vee b_1 \vee b_2) \wedge \bigwedge_{i=0}^{2} (b_i \rightarrow a_i)\,.$$

Such a formula is analogous to those obtained from the encodings of secure state estimation problems discussed in Section IV-D3. We observe that any restriction $\rho_2$ to two of the Boolean variables of $\varphi_2$, for example, $\rho_2(0) = 0$ and $\rho_2(1) = \rho_2(2) = *$, would lead to a formula such as

$$\varphi_{B2} \restriction \rho_2 := (b_1 \vee b_2) \wedge (b_1 \rightarrow a_1) \wedge (b_2 \rightarrow a_2).$$

However, neither $(b_1 \vee b_2) \rightarrow (b_1 \wedge (b_1 \rightarrow b_2))$ nor $(b_1 \vee b_2) \rightarrow (b_2 \wedge (b_2 \rightarrow b_1))$ holds true. Therefore, there are no satisfying assignments and variable orderings that make $\varphi_2$ POM.

A POM formula $\varphi$ can drastically simplify the task of finding a minimal UNSAT certificate. By Definition 6.6, there exists a set of falsifying assignments (implicants) that can be ordered based on their prefixes, and assume the following forms: $\neg b_{\lambda(1)}$, $b_{\lambda(1)} \wedge \neg b_{\lambda(2)}$, $b_{\lambda(1)} \wedge b_{\lambda(2)} \wedge \neg b_{\lambda(3)}$,.... Importantly, because $(\neg a_{\kappa(i)} \rightarrow \neg b_{\lambda(i)})$ holds for all $i \in \{1, \ldots, m\}$, the same prefix structure transfers to the auxiliary variables, so that $\neg a_{\kappa(1)}$, $a_{\kappa(1)} \wedge \neg a_{\kappa(2)}$, $a_{\kappa(1)} \wedge a_{\kappa(2)} \wedge \neg a_{\kappa(3)}$,..., are also falsifying assignments for $\varphi$, and can be used as UNSAT certificates. It is according to this prefix-based ordering of the falsifying implicants that we define a "minimal" UNSAT certificate for $\varphi$. In fact, finding a minimal certificate amounts to looking for the longest prefix associated with a set of consistent convex constraints before an inconsistent constraint is reached according to the variable ordering. We observe that, since we aim to find the earlier occurrence of an inconsistent constraint, a minimal certificate with respect to the prefix order 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 6.1.

We formalize the objective above as follows. Given a POM formula $\varphi$ with respect to $(\mu,\lambda,\kappa)$, let $L$ be the number of variables that are asserted by the valuation $\mu(b, a)$ of SAT-SOLVE, and $a'_\mu = (a'_{\mu,1}, \ldots, a'_{\mu,L})$ be such set, ordered according to $\kappa$. We also denote by $\{(g'_{\mu,1}(x) \lhd 0), \ldots, (g'_{\mu,L}(x) \lhd 0)\}$ the set of convex constraints in $\varphi$ associated with the variables $a'_\mu$. Then, for a constant $\delta \in \mathbb{Q}^+$, we define the function $\textsc{zeroPrefix}_\delta : \mathbb{R}_+^L \rightarrow \mathbb{N}$ as:

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

Intuitively, for small $\delta$, $\textsc{zeroPrefix}_\delta$ returns the first nonzero element of the sequence $s = (s_1, \ldots, s_L)$, and therefore 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 solving the following optimization problem:

**Problem 1:**

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

where $\mathcal{W}$ is the domain of the real variables $x$ and the functions $g'_{\mu,i}$, for all $i$, are defined 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, the solution to Problem 1specifies the longest sequence of convex constraints that are consistent, before an inconsistent constraint is found. 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 nonconvex. It is, however, possible to still find the optimum $k^*$ from Problem 1 using a convex program. To state this result, 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 $\varphi$ feasible. We can define such a bound $\bar{s}$ as follows.

**Definition 6.7:** 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 $\varphi$. We define $\bar{s}$ as the solution of the following convex optimization problem:

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

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



The bound $\bar{s}$ can easily be precomputed offline for a given $\varphi$. 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:**

$$\begin{aligned} \min_{\substack{s_1,\ldots,s_L \in \mathbb{R} \\ x \in \mathcal{W} \subseteq \mathbb{R}^n}} & \sum_{i=1}^{L} |s_i| \\ \text{s.t.} \quad & g'_{\mu,i}(x) \lhd s_i, \quad i = 1, \ldots, L \\ & \frac{\overline{s}}{\delta}\left(\sum_{k=1}^{i-1} |s_k|\right) \leq |s_i|, \quad i = 2, \ldots, L \end{aligned} \tag{29}$$

Problem 2 is a modified version of the SSF problem because of the addition of constraints (29).[2] However, we observe that, if the problem is feasible, constraints (29) 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 $\mu$ is indeed a valid assignment. If, instead, this is not the case, constraints (29) induce an ordering over the nonzero 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, whose proof, for brevity, can be found in the Appendix.

**Theorem 6.8 (Prefix-Based Certificate):** Let $\varphi$ be a POM formula with respect to ($\mu$,$\lambda$,$\kappa$), where $\mu$ is a satisfying assignment for the propositional abstraction $\varphi_B$, and $\varphi$ is defined over a bounded real variable domain $\mathcal{W}$. Let $\delta \in \mathbb{Q}^+$, and $\overline{s} \in \mathbb{R}^+$ be defined as in Definition 6.7, with $\overline{s} \geq \delta$. Let Problem 2 be the feasibility problem associated with $\mu$, $\bar{s}$, and $\delta$. Then, the following hold:

- if Problem 2 is feasible with $\sum_{i=1}^{L} |s_i^*| \leq \delta$ and $x^*$ is the optimum for $x$, then $([\![b]\!]_\mu, x^*) \models \varphi$;
- if Problem 2 is feasible and $k^*$ is the minimum index $k$ such that $\sum_{i=1}^{k} |s_i^*| > \delta$, then, the following clause:

$$\varphi_{POM-ce} := \bigvee_{i=1}^{k^*} \neg a'_{\mu,i}, \tag{30}$$

is an UNSAT certificate which is minimal with respect to $(\mu, \lambda, \kappa)$.

The prefix-based certificate generation procedure can then be implemented as in Algorithm 3. By Theorem 6.8 we can then state the following guarantees of Algorithm 1 with the generation of UNSAT certificates in Algorithm 3.

**Proposition 6.9:** Let $\varphi$ be a monotone SMC formula and $\delta \in \mathbb{Q}^+$ a user-defined tolerance used by $\mathcal{C}$-SOLVE.CHECK and $\mathcal{C}$-SOLVE.PREFIX in Algorithm 1 to accommodate numerical errors. Assume that $\varphi$ is POM for all the satisfying assignments generated by Algorithm 1, so that Algorithm 3 can be used at each iteration of Algorithm 1. Algorithm 1 with the UNSAT certificate from Algorithm 3 is $\delta$-complete.

[2]While constraints (29) are nonconvex, they can be translated into linear constraints using standard transformations dealing with the minimization of the sum of absolute values.

**Algorithm 3:** $(\text{cSTATUS}, x, \varphi_{ce}) = \mathcal{C}\text{-SOLVE.PREFIX}(\mu, \mathcal{M}, \kappa, \delta)$

```
s* := SOLVE-PROBLEM2(μ, M, κ, δ);
if Σ_{i=1}^{L} |s_i^*| ≤ δ then
    cSTATUS = SAT;
    return (cSTATUS, x*, 1)
else
    cSTATUS = UNSAT;
    k* := ZEROPREFIX_δ(s_1^* ... s_L^*);
    φ_ce := ⋁_{i=1}^{k*} ¬a'_{μ,i};
    return (cSTATUS, x*, φ_ce);
end if
```

**Proof (Proposition 6.9):** The proof, along the lines of the one for Proposition 5.1, directly follows from Theorem 6.8.

Overall, as summarized in Table 1, IIS-based certificates are generally the shortest and most effective, but potentially more 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 POM formulas. As also mentioned in Section IV-D, POM formulas can be used to encode the runs of a finite-state transition system. Coupled with continuous dynamics, this pattern arises in several systems, including switched systems, linear hybrid systems, piecewise affine systems, and mixed logical dynamical systems [21].

## VII. RESULTS

We implemented all our algorithms in the prototype solver SatEX [54]. We use Z3[26] as a SAT solver and CPLEX[55] 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 the CPS design problems illustrated in Section III. 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.

### A. Scalability

To test the scalability of our algorithm, we generated SMC problem instances as follows. We used purely Boolean problem instances from the 2014 SAT competition (application track) [56] and selectively included 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 4288 Boolean variables. We then augmented the Boolean instance with

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



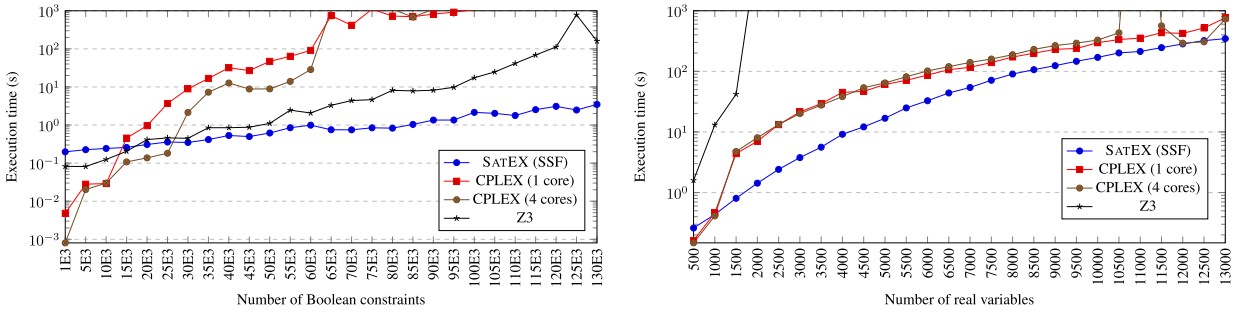



**Fig. 6.** ***Execution time on SMC problem instances, when the number of Boolean constraints increases for a fixed number of 100 real variables (left side) and when the number of real variables increases for a fixed number of 7000 Boolean constraints (right side).***

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. The Boolean instances were certified to be satisfiable while the affine constraints were generated to be feasible, so that SatEX could terminate after one iteration.

Fig. 6 (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 four to five 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 Fig. 6, 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 fast as MIP for problems with a relatively small number of real variables. The gap between SatEX and MIP decreases as the number of real variables increases for a fixed number of Boolean constraints, which is expected, since MIP tends to perform better on problem instances dominated by convex constraints on the reals.

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 affine constraints as above. As shown in Fig. 7, 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.

## B. Application to Secure State Estimation

We apply SatEX to solve the secure state estimation problem illustrated in Section III-C using the formulation in Section IV-D3. We randomly generate the matrices $H_i$ for an increasing number of sensors, randomly select a set of sensors to be under attack and calculate the sensor outputs $Y_i$ using (23), where the value of $\alpha_i$ is also assigned randomly. Fig. 8 compares the performance of SatEX (using the SSF and IIS-based certificates) with the one of a MIP solver (running on two and four cores). 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.

The number of iterations, hence the execution time, of the SMC algorithm changes based on the certificate used. In particular, 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. Overall, the SMC framework allows providing an exact formulation and efficient solution techniques for this combinatorial problem, which was previously mostly tackled using expensive brute force search or approximate convex relaxation approaches.

## C. Application to Motion Planning

The reach-avoid problem examined in Sections III and IV-D2 reduces to a POM formula $\varphi$ for each satisfying assignment $\mu$ of the propositional abstraction $\varphi_B$, as also suggested by the ordering of the Boolean variables

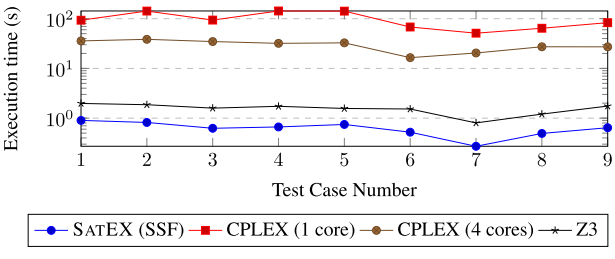



**Fig. 7.** ***Execution time on UNSAT SMC instances due to UNSAT Boolean constraints: the number of Boolean clauses varies from 225 to 960, while the number of real variables is fixed at 500.***

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



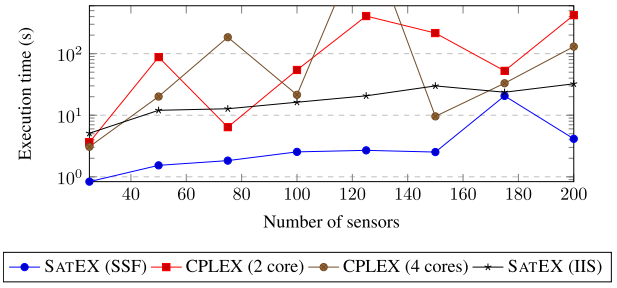



**Fig. 8.** ***Execution time on instances of the secure state estimation problem when the number of sensors increases. Z3 exceeds a 600-s timeout in all benchmarks.***

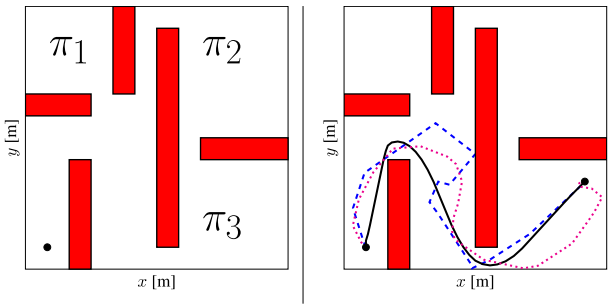



**Fig. 10.** ***(Left) Workspace used in our experiments; (right) trajectories generated by the SMC-based (black), Synergistic RRT (dashed blue), and Synergistic EST (dotted red) motion planners for the double integrator dynamics.***

associated with the different regions according to the transition system in Fig. 5. We can, therefore, exploit our results on prefix-based UNSAT certificates.

*1) Comparison With MIP and SMT Solvers:* Fig. 9 shows the runtime performance of SatEX with respect to a MIP solver on instances of the reach-avoid problem using the formulation in Section IV-D2. We operate with the linearized dynamics of a quadrotor (having 14 continuous states) moving in a 3-D workspace. We partition the workspace into cubes of size 1 m $\times$ 1 m $\times$ 1 m and randomly select some of them to be obstacles. We keep the workspace width and height fixed at 4 m and let its length increase (along the $x$ axis). This translates into increasing both the number of Boolean and continuous variables in $\varphi$, 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. As the $x$-dimension increases, the gap between SatEX and CPLEX increases.

Similarly to our previous experiments, the execution time using both Z3 and dReal exceeds the timeout threshold of 15 minutes and is not shown in Fig. 9.

*2) Comparison With Sampling-Based Motion Planning Techniques:* Sampling-based techniques have become, in the last decade, the most efficient way of solving motion planning problems in robotics. We compare the performance of our SMC-based motion planner with the one of state-of-the-art sampling-based techniques, as implemented in the synergistic combination of layers of planning (SYCLOP) planner, which have shown to outperform traditional sampling-based algorithms by orders of magnitude [57]. SYCLOP, available from the Open Motion Planning Library (OMPL),[3] is a meta-planner that combines a high-level guide computed over a decomposition of the state space with a low-level planning algorithm. The progress that the low-level planner makes is fed back to the high-level planner which uses this information to update the guide. We consider two motion planner versions, namely, SYCLOP RRT and SYCLOP EST using, respectively, the rapidly-exploring random trees (RRTs) and expansive space trees (ESTs) algorithms as their low-level planners.

We consider robot dynamics captured by chains of integrators, one chain for each coordinate of the workspace shown in Fig. 10, and a sampling time of 0.5 s. The robot starts at the point with coordinates $(0.5, 0.5)$ (in meters) and is required to reach the point $(5.5, 2.0)$, while higher order derivatives are set to 0 both at the initial and target points. The upper bound on the control input is $\overline{u} = 0.2$, in appropriate units based on the number of integrators in the chain. Table 2 reports the execution times

[3]https://ompl.kavrakilab.org/planners.html

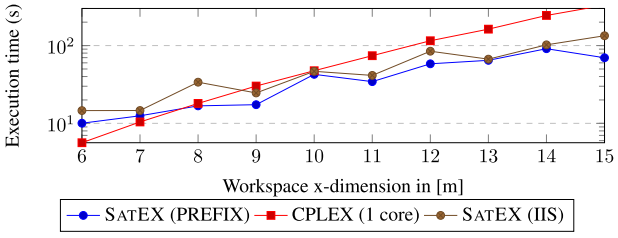



**Fig. 9.** ***Execution time on a set of SMC instances for the motion planning problem as the size of the workspace increases. The execution times for Z3 and dReal exceed the 15-min timeout and are not shown in the figure.***

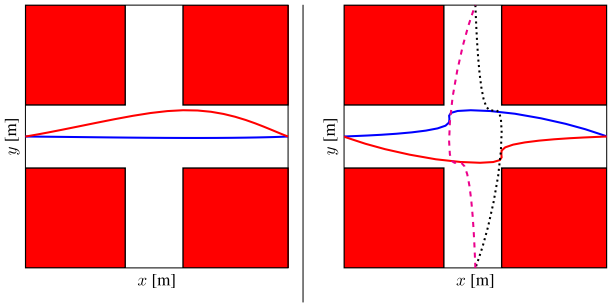



**Fig. 11.** ***Workspace and trajectories under reach avoid specifications for a two-robot scenario (left) and a four-robot scenario (right).***

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



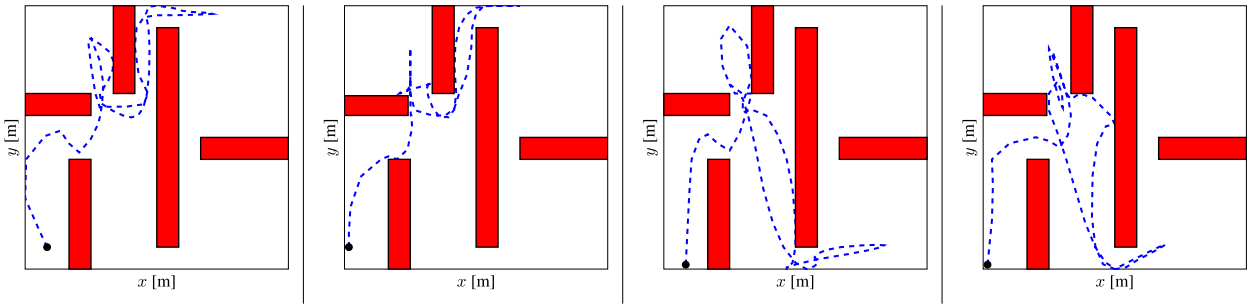



**Fig. 12.** ***Trajectories of robots $R_1$, $R_2$, $R_3$, and $R_4$ (from left to right), subject to the following specification: "Infinitely often all robots shall simultaneously visit region $\pi_1$; moreover, infinitely often, two robots shall simultaneously visit region $\pi_2$ while the other two robots shall simultaneously visit region $\pi_3$." The trajectories of $R_1$ and $R_2$ visit region $\pi_2$ while the ones of $R_3$ and $R_4$ touch region $\pi_3$ as specified.***

of different algorithms as the number of integrators in the chain, hence the number of state variables, increases. Times are averaged over 20 trials. RRT and EST-based planners show much higher variability in execution time than the SMC-based planner, as is expected because of their randomized search schemes. SYCLOP EST performs better for a small number of continuous states, but its runtime rapidly increases and reaches a 1-hour timeout for a chain of four integrators. Our algorithm scales better over the whole range of continuous states scoring more than one order of magnitude reduction in computation time. Moreover, the generated trajectory, as shown in Fig. 10, is usually smoother.

*3) Multirobot Motion Planning:* As discussed in Section IV-D2, the motion planning problem for a team of robots, required to achieve a set of goals under collision avoidance constraints, can also be translated into the satisfiability problem for a monotone SMC formula. We first show the effectiveness of the encoding scheme on the workspace in Fig. 11, where the robots are required to traverse the same workspace region as they move from their initial positions to their targets, subject to reach-avoid specifications. Table 3 reports the performance of our motion planner as the number of robots (hence the number of Boolean variables in the problem) and the number of integrators (hence the continuous states in the problem) increase. Trajectories for a two-robot and a four-robot scenario are visualized, respectively, on the left and right sides of Fig. 11, illustrating the satisfaction of the collision avoidance constraints with a safety margin $\epsilon = 0.2$ m.

We then demonstrate the capabilities of our framework on a multirobot scenario subject to more complex high-level specifications. We consider the same workspace in Fig. 10, which contains three regions, denoted by $\pi_1, \pi_2$ and $\pi_3$, a team of four robots, and the following specification: "Infinitely often all robots shall simultaneously visit region $\pi_1$; moreover, infinitely often, two robots shall simultaneously visit region $\pi_2$ while the other two robots shall simultaneously visit region $\pi_3$." This specification can be captured by an LTL formula that can be then encoded into a set of Boolean constraints over a fixed time horizon [36]. We report in Table 3 the performance of our motion planner as the number of robots and chained integrators increase, while being subject to a similar specification as the one above, together with the problem size in terms of number of Boolean and real variables. The trajectories for the four-robot scenario are separately shown in Fig. 12.

## D. Application to an ARPOD Mission

We finally apply our algorithm to the ARPOD test-case introduced in Section III-A, following the formulation

**Table 2** Execution Time of Different Motion Planning Algorithms as Function of Number of Continuous States for Workspace in Fig. 10. Results Are Averaged Across 20 trials. Timeout Is Set to 1 h

| Number of States | SMC-Based [s] | Synergistic RRT [s] | Synergistic EST [s] |
|---|---|---|---|
| 4 | 3.007 | 33.166 | 0.6151 |
| 6 | 4.590 | 3216.402 | 791.444 |
| 8 | 7.502 | timeout | timeout |
| 10 | 10.207 | timeout | timeout |
| 12 | 34.775 | timeout | timeout |
| 14 | 60.413 | timeout | timeout |
| 16 | 39.070 | timeout | timeout |
| 18 | 70.631 | timeout | timeout |
| 20 | 75.843 | timeout | timeout |

**Table 3** Performance of SMC-based Motion Planner and Size of the Problem in Multirobot Scenarios with Reach-Avoid Specification as Well as More Complex High-Level specification. Timeout Is Set to 30 min

| Number of Robots | Number of States (per robot) | SMC-Based Reach-avoid specification | | | SMC-Based High-level complex specification | | |
|---|---|---|---|---|---|---|---|
| | | time [s] | #real vars | #Boolean vars | time [s] | #real vars | #Boolean vars |
| 2 | 4 | 0.3346 | 336 | 66 | 19.269 | 960 | 2370 |
| | 6 | 0.822 | 420 | | 44.72625 | 1200 | |
| | 8 | 1.0625 | 504 | | 67.6701 | 1440 | |
| | 10 | 0.915 | 588 | | 76.3877 | 1680 | |
| | 12 | 2.444 | 672 | | 665.4057 | 1920 | |
| 3 | 4 | 0.7170 | 504 | 117 | 105.661 | 1440 | 3449 |
| | 6 | 2.1074 | 630 | | 196.425 | 1800 | |
| | 8 | 3.8263 | 756 | | 253.077 | 2160 | |
| | 10 | 15.005 | 882 | | 1151.087 | 2520 | |
| | 12 | 8.654 | 1008 | | 466.6257 | 2880 | |
| 4 | 4 | 0.9621 | 672 | 180 | 444.354 | 1920 | 4648 |
| | 6 | 5.1138 | 840 | | 829.665 | 2400 | |
| | 8 | 6.3493 | 1008 | | 986.9366 | 2880 | |
| | 10 | 44.4658 | 1176 | | timeout | 3360 | |
| | 12 | 80.0632 | 1344 | | timeout | 3840 | |
| 5 | 4 | 5.8121 | 840 | 255 | 1334.822 | 2400 | 5967 |
| 6 | 4 | 26.4051 | 1008 | 342 | timeout | 2880 | 7406 |
| 7 | 4 | 142.896 | 1008 | 441 | timeout | 3360 | 8965 |
| 8 | 4 | 1229.5425 | 1344 | 552 | timeout | 3840 | 10644 |

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



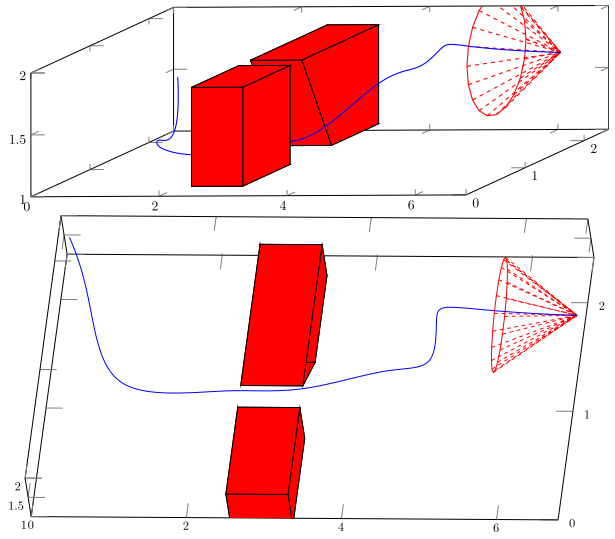


**Fig. 13.** ***Trajectory generated by SatEX for the ARPOD test-case (all units are in kilometers). The different views show the shape of the obstacles and the LOS cone (top) as well as the position of the start and docking points (bottom).***

in Section IV-D1. In our experiments, we also introduce a set of obstacles in the rendezvous phase, which must be avoided by the spacecraft. Fig. 13 shows the final spacecraft trajectory in the rendezvous and docking phases from two different angles. As shown at the top of Fig. 13, the spacecraft first lowers its altitude to avoid the obstacles and then increases it again until it reaches the center of the LOS cone opening. At this point, the docking phase starts, and a higher sampling rate is used for the spacecraft trajectory. The spacecraft navigates within the LOS cone until it reaches the docking point with zero velocity, when the docked phase starts. The trajectory was generated in 3.2 minutes.

## VIII. CONCLUSION

In this paper, we revisited the connection between Boolean methods and convex programming toward a novel, scalable framework for reasoning about the combination of discrete and continuous dynamics that can address the complexity of cyber–physical system applications. We introduced a procedure for determining the satisfiability of a class of logic formulas over Boolean variables and convex constraints, termed monotone SMC formulas, that appear in the formulation of estimation and control design problems arising in different contexts. We showed that SMC formulas are the most general class of formulas over Boolean and nonlinear real predicates that can be solved via a finite number of convex programs. For these formulas, we proposed a lazy combination of satisfiability (SAT) solving and convex programming, namely, satisfiability modulo convex programming, to provide a satisfying assignment or determine that the formula is unsatisfiable. 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 satisfiability modulo theory (SMT) and mixed integer convex programming (MICP) solvers on problems with complex Boolean structure and a large number of real variables. The proposed SMC scheme can then be used to build effective and scalable decision procedures for a wide variety of problems including the verification and control of cyber–physical systems. ■

## APPENDIX

**Proof of Theorem 6.8:** To prove Theorem 6.8 it is enough to show the following result.

**Theorem 8.1:** Let $\overline{s} \in \mathbb{R}^+$, as in Definition 6.7, and $\delta \in \mathbb{Q}^+$ satisfy $\overline{s} \geq \delta$. Then, an optimal solution of Problem 2 is also an optimal solution of Problem 1.

We first state and prove an intermediate result that is used to prove Theorem 8.1.

**Proposition 8.2:** For a valid assignment $\mu$ from SAT-SOLVE, let $(a'_{\mu,1}, \ldots, a'_{\mu,L})$ be the set of variables that are asserted, ordered according to the ordering $\kappa$, with respect to which $\varphi$ is a POM formula. Let the variables $s_1, \ldots, s_L$ be selected as the solution of Problem 2, i.e., to minimize $\sum_{i=1}^{L} |s_i|$ such that the following constraints are satisfied:

$$g'_{\mu,i}(x) \lhd s_i, \qquad i = 1, \ldots, L \tag{31}$$

$$\frac{\overline{s}}{\delta}\left(\sum_{k=1}^{i-1} |s_k|\right) \leq |s_i|, \qquad i = 2, \ldots, L \tag{32}$$

where $g'_{\mu,i}(x)$, $\delta \in \mathbb{Q}^+$, and $\overline{s}$ are defined as in Section VI-C, and $\overline{s} \geq \delta$ holds. The following results hold:

- Assume $|s_j| > 0$ for some $j \in \{1, \ldots, L\}$. Then, for all $j' \in \{1, \ldots, L\}$, $j' > j$, we obtain

$$|s_{j'}| \geq O_{\overline{s},\delta}(j' - j)\left(\sum_{k=1}^{j} |s_k|\right) \tag{33}$$

  where $O_{\overline{s},\delta}(j' - j)$ is a constant that depends only on $j' - j$.
- Assume further that $|s_j| > \delta$ for some $j \in \{1, \ldots, L\}$. Then, the constraints in (33) hold as equalities, i.e., for all $j' \in \{1, \ldots, L\}$, $j' > j$, we obtain

$$|s_{j'}| = O_{\overline{s},\delta}(j' - j)\left(\sum_{k=1}^{j} |s_k|\right). \tag{34}$$

*Proof:* If $|s_j| > 0$ holds and constraints (32) are satisfied, it is straightforward to show, e.g., by induction, that the following inequality holds for all $j' > j$:

$$|s_{j'}| \geq \frac{\overline{s}}{\delta}\left(\sum_{k=1}^{j} |s_k|\right)\left(1 + \frac{\overline{s}}{\delta}\right)^{j'-j-1} \tag{35}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



which leads to (33) after setting $O_{\overline{s},\delta}(j'-j) = (\overline{s}/\delta)(1+(\overline{s}/\delta))^{j'-j-1}$.

To prove (34) for all $j' > j$, assume $|s_j| > \delta$. Then, (35) implies $|s_{j'}| \geq (\overline{s}/\delta)(\sum_{k=1}^{j}|s_k|) \geq (\overline{s}/\delta)|s_j| > \overline{s}$. Any solution of Problem 2 under the assumption $|s_j| > \delta$ would then produce slack variables $|s_{j'}|$ larger than $\overline{s}$ for all $j' > j$. On the other hand, since $\overline{s}$ is an upper bound on the minimum slacks that make all convex constraints consistent over all $x \in \mathcal{W}$, we also observe that a sequence of slack values such that $|s_i| \leq \overline{s}$ for all $i$ would be enough to satisfy all the constraints in Problem 2 except for constraints (32). Therefore, as Problem 2 attempts to minimize the sum of the slacks subject to constraints (32), all the slack variables $|s_{j'}|$ with $j' > j$ will be pushed towards their lower bounds in (32). The subset of constraints (32) with $i \in \{j+1,\dots,L\}$ will then be active at optimum, i.e., they will hold as equality constraints. Finally, at optimum, (35) will also turn into equality by the same argument, thus leading to

$$\begin{aligned}|s_{j'}| &= \frac{\overline{s}}{\delta}\left(\sum_{k=1}^{j}|s_k|\right)\left(1+\frac{\overline{s}}{\delta}\right)^{j'-j-1}\\ &= O_{\overline{s},\delta}(j'-j)\left(\sum_{k=1}^{j}|s_k|\right)\end{aligned} \tag{36}$$

for all $j' > j$, which is what we wanted to prove. □

We are now ready to prove Theorem 8.1.

**Proof (Theorem 8.1):** As a first step, we consider the following optimization problem in the context of our formulation.

**Problem 3:**

$$\begin{aligned}\max_{\substack{s_1,\dots,s_L\in\mathbb{R}\\ x\in\mathcal{W}\subseteq\mathbb{R}^n}}\ & \textsc{zeroPrefix}_\delta(s_1,\dots,s_L)\\ \text{s.t. } & g'_{\mu,i}(x) \lhd s_i, \quad i=1,\dots,L\\ & \frac{\overline{s}}{\delta}\left(\sum_{k=1}^{i-1}|s_k|\right) \leq |s_i| \quad i=2,\dots,L\end{aligned} \tag{37}$$

Problem 3 is a constrained version of Problem 1, due to the introduction of the constraints (37). However, for any optimal solution $(\tilde{s}_1,\dots,\tilde{s}_L)$ of Problem 1 of the form $(|\tilde{s}_1|,\dots,|\tilde{s}_{j-1}|,|\tilde{s}_j|,|\tilde{s}_{j+1}|,\dots,|\tilde{s}_L|)$ with $\sum_{k=1}^{j-1}|\tilde{s}_k| \leq \delta$ and $|\tilde{s}_j| > \delta$, we can always construct an optimal solution $(s_1,\dots,s_L)$ of Problem 3 of the form $(|s_1|,\dots,|s_{j-1}|,|s_j|,|s_{j+1}|,\dots,|s_L|)$ and such that $|s_k| = |\tilde{s}_k|$ for all $k=1,\dots,j$, and $|s_{j'}|$ satisfies

$$|s_{j'}| = O_{\overline{s},\delta}(j'-j)\left(|s_j| + \sum_{k=1}^{j-1}|s_k|\right) = O_{\overline{s},\delta}(j'-j)|s_j|,$$

for all $j'$ such that $j+1 \leq j' \leq L$, $O_{\overline{s},\delta}(j'-j)$ being the constant defined as in Proposition 8.2 in the Appendix. Therefore, the maximum of Problem 1 is also achieved by Problem 3, and solving Problem 3 is enough to retrieve it.

To prove our final result, it is then enough to show that a solution of Problem 2 is also a solution of Problem 3. To do so, we proceed by contradiction. Let

$$s^* = (s_1^*,\dots,s_L^*), \quad i=0,\dots,L$$

be an optimal solution of Problem 2. We then assume that $s^*$ is not a solution of Problem 3, i.e., there exists $s=(s_1,\dots,s_L)$ such that

$$\textsc{zeroPrefix}_\delta(s_1^*,\dots,s_L^*) < \textsc{zeroPrefix}_\delta(s_1,\dots,s_L) \tag{38}$$

or, equivalently

$$\textsc{zeroPrefix}_\delta(s_1^*,\dots,s_L^*) + 1 \leq \textsc{zeroPrefix}_\delta(s_1,\dots,s_L). \tag{39}$$

Given $j = \textsc{zeroPrefix}_\delta(s^*)$, by definition of $\textsc{zeroPrefix}_\delta$ and by (39), we obtain

$$\sum_{i=1}^{j}|s_i^*| > \delta, \qquad \sum_{i=1}^{j}|s_i| \leq \delta. \tag{40}$$

Moreover, by (39) and the definition of $\overline{s}$, we can state, without loss of generality, that $s$ satisfies the following properties:

$$0 \leq |s_{j+1}| \leq \overline{s} \quad \forall\, i \in \{j+2,\dots,L\} \tag{41}$$

$$0 \leq |s_i| \leq O_{\overline{s},\delta}(i-j-1)\left(\sum_{k=1}^{j}|s_k| + \overline{s}\right) \tag{42}$$

where the upper bound in (42) is obtained by using the result in Proposition 8.2 and (41).

We compute now the cost function of Problem 2 for both $s^*$ and $s$. In the first case, we obtain

$$\begin{aligned}\sum_{i=1}^{L}|s_i^*| &= \sum_{i=1}^{j}|s_i^*| + |s_{j+1}^*| + \sum_{i=j+2}^{L}|s_i^*|\\ &\overset{(a)}{>} \delta + |s_{j+1}^*| + \sum_{i=j+2}^{L}|s_i^*|\\ &\overset{(b)}{>} \delta + \overline{s} + \sum_{i=j+2}^{L}|s_i^*|\\ &\overset{(c)}{=} \delta + \overline{s} + \sum_{i=j+2}^{L} O_{\overline{s},\delta}(i-j-1)\left(\sum_{k=1}^{j}|s_k^*| + |s_{j+1}^*|\right)\\ &\overset{(d)}{>} \delta + \overline{s} + \sum_{i=j+2}^{L} O_{\overline{s},\delta}(i-j-1)\,(\delta+\overline{s})\end{aligned} \tag{43}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



where $(a)$ follows from (40), $(b)$ follows from (29), which implies that $|s^*_{j+1}| \geq (\overline{s}/\delta)(\sum_{i=0}^{j}|s^*_i|) > (\overline{s}/\delta)\cdot\delta = \overline{s}$, $(c)$ follows from Proposition 8.2 and the assumption that $\overline{s} \geq \delta$, and $(d)$ follows again from (40) and (29). On the other hand, we also obtain

$$\begin{aligned}\sum_{i=1}^{L}|s_i| &= \sum_{i=1}^{j}|s_i| + |s_{j+1}| + \sum_{i=j+2}^{L}|s_i| \\ &\overset{(e)}{\leq} \delta + |s_{j+1}| + \sum_{i=j+2}^{L}|s_i| \\ &\overset{(f)}{\leq} \delta + \overline{s} + \sum_{i=j+2}^{L} O_{\overline{s},\delta}(i-j-1)\left(\sum_{k=1}^{j}|s_k| + \overline{s}\right)\end{aligned}$$

$$\overset{(g)}{\leq} \delta + \overline{s} + \sum_{i=j+2}^{L} O_{\overline{s},\delta}(i-j-1)\,(\delta+\overline{s}) \tag{44}$$

where $(e)$ follows from (40), $(f)$ follows from (41) and (42), and $(g)$ follows from (40).

From both (44) and (43), we finally conclude

$$\sum_{i=1}^{L}|s^*_i| > \sum_{i=1}^{L}|s_i| \tag{45}$$

stating that $s^*$ is not a minimal point for the objective function of Problem 2, which is in contradiction with the initial assumption of $s^*$ being an optimal solution. □

## REFERENCES


[1] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. Cambridge, MA, USA: MIT Press, 2016.

[2] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-based design for cyber-physical systems," *Eur. J. Control*, vol. 18, no. 3, pp. 217–238, 2012.

[3] J. Sifakis, "Rigorous system design," *Found. Trends Electron. Design Autom.*, vol. 6, no. 4, pp. 293–362, 2013, doi: doic 10.1561/1000000034.

[4] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proc. IEEE*, vol. 103, no. 11, pp. 2104–2132, Nov. 2015.

[5] P. Nuzzo, "A contract-based methodology for aircraft electric power system design," *IEEE Access*, vol. 2, pp. 1–25, 2014.

[6] S. A. Seshia, "Combining induction, deduction, and structure for verification and synthesis," *Proc. IEEE*, vol. 103, no. 11, pp. 2036–2051, Nov. 2015.

[7] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Trans. Autom. Control*, vol. 51, no. 12, pp. 1862–1877, Dec. 2006.

[8] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 53, no. 1, pp. 287–297, Feb. 2008.

[9] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, Feb. 2009.

[10] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans. Robot.*, vol. 25, no. 6, pp. 1370–1381, Dec. 2009.

[11] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Autom. Control*, vol. 57, no. 11, pp. 2817–2830, Nov. 2012.

[12] M. Rungger, M. Mazo, Jr., and P. Tabuada, "Specification-guided controller synthesis for linear systems and safe linear-time temporal logic," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2013, pp. 333–342.

[13] S. Malik and L. Zhang, "Boolean satisfiability: From theoretical hardness to practical success," *Commun. ACM*, vol. 52, no. 8, pp. 76–82, 2009.

[14] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[15] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Satisfiability*. Amsterdam, The Netherlands: IOS Press, 2009.

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

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

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

[19] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-UAV mission planning," *Int. J. Robust Nonlinear Control*, vol. 21, no. 12, pp. 1372–1395, 2011.

[20] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *Proc. IEEE Int. Conf. Robot. Autom.*, May/Jun. 2014, pp. 5319–5325.

[21] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, Mar. 1999.

[22] 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.

[23] A. Cimatti, A. Franzén, A. Griggio, R. Sebastiani, and C. Stenico, "Satisfiability modulo the theory of costs: Foundations and applications," in *Proc. Int. Conf. Tools Algorithms Construct. Anal. Syst.*, 2010, pp. 99–113.

[24] Y. Li, A. Albarghouthi, Z. Kincaid, A. Gurfinkel, and M. Chechik, "Symbolic optimization with SMT solvers," *ACM SIGPLAN Notices*, vol. 49, no. 1, pp. 607–618, 2014.

[25] A. Bauer, M. Pister, and M. Tautschnig, "Tool-support for the analysis of hybrid systems and models," in *Proc. DATE*, 2007, pp. 1–6.

[26] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proc. Int. Conf. Tools Algorithms Construct. Anal. Syst.*, 2008, pp. 337–340.

[27] D. Jovanović and L. de Moura, "Solving nonlinear arithmetic," in *Proc. Int. Joint Conf. Autom. Reasoning*, 2012, pp. 339–354.

[28] M. Franzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige, "Efficient solving of large nonlinear arithmetic constraint systems with complex Boolean structure," in *Proc. JSAT Special Issue SAT/CP Integr.*, 2007.

[29] S. Gao, S. Kong, and E. M. Clarke, "dReal: An SMT solver for nonlinear theories over the reals," in *Proc. Int. Conf. Autom. Deduct.*, vol. 7898, 2013, pp. 208–214.

[30] P. Nuzzo, A. Puggelli, S. A. Seshia, and A. Sangiovanni-Vincentelli, "CalCS: SMT solving for non-linear convex constraints," in *Proc. Formal Methods Computer-Aided Design*, Oct. 2010, pp. 71–79.

[31] Y. Shoukry, P. Nuzzo, A. Puggelli, A. L. Sangiovanni-Vincentelli, S. A. Seshia, and P. Tabuada, "Secure state estimation for cyber-physical systems under sensor attacks: A satisfiability modulo theory approach," *IEEE Trans. Autom. Control*, vol. 62, no. 10, pp. 4917–4932, Oct. 2017.

[32] Y. Shoukry, "IMHOTEP-SMT: A satisfiability modulo theory solver for secure state estimation," in *Proc. Int. Workshop Satisfiability Modulo Theories*, Jul. 2015, pp. 3–13.

[33] Y. Shoukry, A. Puggelli, P. Nuzzo, A. Sangiovanni-Vincentelli, S. Seshia, and P. Tabuada, "Sound and complete state estimation for linear dynamical systems under sensor attacks using Satisfiability Modulo Theory solving," in *Proc. Amer. Control Conf.*, Jul. 2015, pp. 3818–3823.

[34] Y. Shoukry, P. Nuzzo, N. Bezzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, and P. Tabuada, "Secure state reconstruction in differentially flat systems under sensor attacks using satisfiability modulo theory solving," in *Proc. Int. Conf. Decision Control*, Dec. 2015, pp. 3804–3809.

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

[36] Y. Shoukry, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *Proc. Int. Conf. Decision Control*, Dec. 2017, pp. 1132–1137.

[37] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "SMC: Satisfiability modulo convex optimization," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2017, pp. 19–28.

[38] C. Jewison and R. S. Erwin, "A spacecraft benchmark problem for hybrid control and estimation," in *Proc. Int. Conf. Decision Control*, Dec. 2016, pp. 3300–3305.

[39] M. Fränzle and C. Herde, "HySAT: An efficient proof engine for bounded model checking of hybrid systems," *Formal Methods Syst. Design*, vol. 30, no. 3, pp. 179–198, 2007.

[40] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI Commun.*, vol. 29, no. 1, pp. 151–162, 2016.

[41] A. Pnueli, "The temporal logic of programs," in *Proc. FOCS*, Oct./Nov. 1977, pp. 46–57.

[42] M. Grant, S. Boyd, and Y. Ye, "Disciplined convex programming," in *Global Optimization*. New York, NY, USA: Springer-Verlag, 2006, pp. 155–210.

[43] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*. New York, NY, USA: Springer-Verlag, 2008.

[44] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, 1995.

[45] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. Philadelphia, PA, USA: SIAM, 2002, vol. 80.

[46] H. Woźniakowski, "Roundoff-error analysis of a new class of conjugate-gradient algorithms," *Linear Algebra Appl.*, vol. 29, pp. 507–529, Feb. 1980.

[47] A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan, "Linear encodings of bounded LTL model checking," *Logical Methods Comput. Sci.*, vol. 2, no. 5, pp. 1–64, 2006.

[48] J. Ding, E. Li, H. Huang, and C. J. Tomlin, "Reachability-based synthesis of feedback policies

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



for motion planning under bounded disturbances," in *Proc. Int. Conf. Robot. Autom.*, May 2011, pp. 2160–2165.
[49] K. Margellos and J. Lygeros, "Hamilton–Jacobi formulation for reach–avoid differential games," *IEEE Trans. Autom. Control*, vol. 56, no. 8, pp. 1849–1861, Aug. 2011.
[50] Z. Zhou, R. Takei, H. Huang, and C. J. Tomlin, "A general, open-loop formulation for reach-avoid games," in *Proc. Conf. Decision Control*, Dec. 2012, pp. 6501–6506.
[51] J. W. Chinneck and E. W. Dravnieks, "Locating minimal infeasible constraint sets in linear programs," *ORSA J. Comput.*, vol. 3, no. 2, pp. 157–168, 1991.
[52] U. Junker, "QUICKXPLAIN: Preferred explanations and relaxations for over-constrained problems," in *Proc. AAAI*, 2004, pp. 167–172.
[53] E. Amaldi, M. E. Pfetsch, and L. E. Trotter, "Some structural and algorithmic properties of the maximum feasible subsystem problem," in *Integer Programming and Combinatorial Optimization*, G. Cornuéjols, R. E. Burkard, and G. J. Woeginger, Eds. Berlin, Germany: Springer, 1999, pp. 45–59.
[54] (Jun. 2018). *SatEX Solver*. [Online]. Available: https://yshoukry.bitbucket.io/SatEX/
[55] (Feb. 2012). *IBM ILOG CPLEX Optimizer*. [Online]. Available: http://www.ibm.com/software/integration/optimization/cplex-optimizer/
[56] The International SAT Competitions Web Page. Accessed: Oct. 1, 2016. [Online]. Available: http://www.satcompetition.org/
[57] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 469–482, Jun. 2010.

## ABOUT THE AUTHORS



**Yasser Shoukry** (Member, IEEE) received the B.Sc. and the M.Sc. degrees (with distinction and honors) in computer and systems engineering from Ain Shams University, Cairo, Egypt, in 2007 and 2010, respectively, and the Ph.D. degree in electrical engineering from the University of California at Los Angeles (UCLA), Los Angeles, CA, USA, in 2015, where he was affiliated with both the Cyber-Physical Systems Lab as well as the Networked and Embedded Systems Lab.

He joined the University of Maryland, College Park, MD, USA, in 2017 as an Assistant Professor in the Department of Electrical and Computer Engineering where he leads the Resilient Cyber-Physical Systems Laboratory. Between September 2015 and July 2017, he was a joint Postdoctoral Researcher at the University of California, Berkeley, the University of California, Los Angeles, and the University of Pennsylvania. Before pursuing his Ph.D. at UCLA, he spent four years as an R&D engineer in the industry of automotive embedded systems. His current research interestes include the design and implementation of resilient cyber–physical systems by drawing on tools from embedded systems, formal methods, control theory, and machine learning.

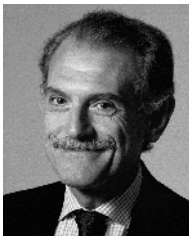
Dr. Shoukry is the recipient of the Best Demo Award from the ACM/IEEE IPSN conference in 2017, the Best Paper Award from the ACM/IEEE ICCPS in 2016, the Distinguished Dissertation Award from the UCLA Electrical Engineering Department in 2016 and the UCLA Chancellor's prize in 2011/2012. In 2015, he led the UCLA/Caltech/CMU team to win the NSF Early Career Investigators (NSF-ECI) research challenge. His team represented the NSF-ECI in the NIST Global Cities Technology Challenge, an initiative designed to advance the deployment of Internet of Things (IoT) technologies within a smart city.



**Pierluigi Nuzzo** (Member, IEEE) received the Laurea degree in electrical engineering (*summa cum laude*) from the University of Pisa, Italy, in 2003, the Diploma in engineering (*summa cum laude*) from the Sant'Anna School of Advanced Studies, Pisa, Italy, in 2004, and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 2015.

He joined the University of Southern California, Los Angeles, CA, USA, in 2016, as an Assistant Professor in the Department of Electrical Engineering. Before joining the University of California at Berkeley, he was a Researcher at IMEC, Leuven, Belgium, working on the design of energy-efficient A/D converters and frequency synthesizers for reconfigurable radio. From 2004 to 2006, he was with the Department of Information Engineering, University of Pisa, and with IMEC, as a Visiting Scholar, working on low power A/D converter design for wide-band communications and design methodologies for mixed-signal integrated circuits. His current research interests include: methodologies and tools for cyber-physical system and mixed-signal system design; contracts, interfaces and compositional methods for embedded system design; the application of formal methods and optimization theory to problems in embedded and cyber-physical systems and electronic design automation.

Prof. Nuzzo received First Place in the operational category and Best Overall Submission in the 2006 DAC/ISSCC Design Competition, a Marie Curie Fellowship from the European Union in 2006, the University of California at Berkeley EECS Departmental Fellowship in 2008, the University of California at Berkeley Outstanding Graduate Student Instructor Award in 2013, the IBM Ph.D. Fellowship in 2012 and 2014, the Best Paper Award from the International Conference on Cyber-Physical Systems (ICCPS) in 2016, and the David J. Sakrison Memorial Prize in 2016.



**Alberto L. Sangiovanni-Vincentelli** (Fellow, IEEE) holds the Buttner Chair of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA. On the industrial side, he helped founding Cadence and Synopsys, the two leading companies in Electronic Design Automation and is on the Board of five companies including Cadence. He is on the Advisory Board of three companies and has consulted for companies such as Intel, HP, Bell Labs, IBM, Samsung, UTC, Kawasaki Steel, Fujitsu, Telecom Italia, Pirelli, BMW, Mercedes, Magneti Marelli, ST Microelectronics, LElettronica and UniCredit. He is an author of over 900 papers, 17 books and two patents.

Dr. Sangiovanni-Vincentelli was awarded the IEEE/RSE James Clerk Maxwell Award for "groundbreaking contributions that have had an exceptional impact on the development of electronics and electrical engineering or related fields," the Kaufmann Award for seminal contributions to EDA, the IEEE Darlington Award, the IEEE Guillemin-Cauer Award, the EDAA lifetime Achievement Award, the IEEE/ACM R. Newton Impact Award, the University of California Distinguished Teaching Award, the SRC Aristotle Award and the IEEE Graduate Teaching Award for inspirational teaching of graduate students. He is a fellow of the IEEE and the ACM, a member of the National Academy of Engineering, and holds two honorary Doctorates.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.





**Sanjit A. Seshia** (Fellow, IEEE) received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Bombay, India, in 1998 and the M.S. and Ph.D. degrees in computer science from Carnegie Mellon University, Pittsburgh, PA, USA, in 2000 and 2005, respectively.

He is currently a Professor in the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA. His research interests are in dependable computing and computational logic, with a current focus on applying automated formal methods to problems in embedded and cyber–physical systems, electronic design automation, computer security, and artificial intelligence. His Ph.D. thesis work on the UCLID verifier and decision procedure helped pioneer the area of satisfiability modulo theories (SMT) and SMT-based verification. He is co-author of a widely used textbook on embedded systems. He led the offering of a massive open online course on cyber-physical systems for which his group developed novel virtual lab auto-grading technology based on formal methods.

Prof. Seshia has served as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, and as Co-Chair of the Program Committee of the International Conference on Computer-Aided Verification (CAV) in 2012. His awards and honors include a Presidential Early Career Award for Scientists and Engineers (PECASE) from the White House, an Alfred P. Sloan Research Fellowship, the Frederick Emmons Terman Award for engineering education, and the School of Computer Science Distinguished Dissertation Award at Carnegie Mellon University.



**George J. Pappas** (Fellow, IEEE) received the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 1998.

He is currently the Joseph Moore Professor and Chair of the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA. He also holds a secondary appointment with the Department of Computer and Information Sciences and the Department of Mechanical Engineering and Applied Mechanics. He is a Member of the GRASP Lab and the PRECISE Center. He had previously served as the Deputy Dean for Research with the School of Engineering and Applied Science. His research interests include control theory and, in particular, hybrid systems, embedded systems, cyberphysical systems, and hierarchical and distributed control systems, with applications to unmanned aerial vehicles, distributed robotics, green buildings, and biomolecular networks.

Dr. Pappas has received various awards, such as the Antonio Ruberti Young Researcher Prize, the George S. Axelby Award, the Hugo Schuck Best Paper Award, the George H. Heilmeier Award, the National Science Foundation PECASE award and numerous best student papers awards.



**Paulo Tabuada** (Fellow, IEEE) was born in Lisbon, Portugal, one year after the Carnation Revolution. He received the "Licenciatura" degree in aerospace engineering from the Instituto Superior Tecnico, Lisbon, Portugal, in 1998 and the Ph.D. degree in electrical and computer engineering from the Institute for Systems and Robotics, a private research institute associated with Instituto Superior Tecnico, in 2002.

Between January 2002 and July 2003, he was a Postdoctoral Researcher at the University of Pennsylvania. After spending three years at the University of Notre Dame, as an Assistant Professor, he joined the Electrical and Computer Engineering Department, University of California at Los Angeles, Los Angeles, CA, USA, where he currently is the Vijay K. Dhir Professor of Engineering.

Dr. Tabuada received multiple awards including the NSF CAREER award in 2005, the Donald P. Eckman award in 2009, the George S. Axelby award in 2011, and the Antonio Ruberti Prize in 2015. In 2009, he co-chaired the International Conference Hybrid Systems: Computation and Control (HSCC'09) and joined its steering committee in 2015; in 2012, he was Program Co-Chair for the 3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys'12); in 2015, he was Program Co-Chair for the IFAC Conference on Analysis and Design of Hybrid Systems; and in 2018, he was Program Co-Chair for the International Conference on Cyber-Physical Systems (ICCPS'18). He also served on the editorial board of IEEE EMBEDDED SYSTEMS LETTERS and the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.