CalCS: SMT Solving for Non-Linear Convex Constraints

Pierluigi Nuzzo, Alberto Puggelli, Sanjit A. Seshia and Alberto Sangiovanni-Vincentelli
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
Berkeley, California 94720
Email: {nuzzo, puggelli, sseshia, alberto}@eecs.berkeley.edu

***Abstract*—Certain formal verification tasks require reasoning about Boolean combinations of non-linear arithmetic constraints over the real numbers. In this paper, we present a new technique for satisfiability solving of Boolean combinations of non-linear constraints that are convex. Our approach applies fundamental results from the theory of convex programming to realize a satisfiability modulo theory (SMT) solver. Our solver, CalCS, uses a lazy combination of SAT and a theory solver. A key step in our algorithm is the use of complementary slackness and duality theory to generate succinct infeasibility proofs that support conflict-driven learning. Moreover, whenever non-convex constraints are produced from Boolean reasoning, we provide a procedure that generates conservative approximations of the original set of constraints by using geometric properties of convex sets and supporting hyperplanes. We validate CalCS on several benchmarks including formulas generated from bounded model checking of hybrid automata and static analysis of floating-point software.**

I. Introduction

The design and verification of certain systems requires reasoning about nonlinear equalities and inequalities, both algebraic and differential. Examples range from mixed-signal integrated circuits (e.g., [1]) that should operate correctly over process-voltage-temperature variations, to control design for biological or avionics systems, for which safety must be enforced (e.g., [2]). In order to extend the reach of formal verification methods such as bounded model checking (BMC) for such systems [3], [4], it is necessary to develop efficient satisfiability modulo theories (SMT) solvers [5] for Boolean combinations of non-linear arithmetic constraints. However, SMT solving for arbitrary non-linear arithmetic over the reals, involving, e.g., quantifiers and transcendental functions, is undecidable [6]. There is therefore a need to develop efficient solvers for special cases that are also useful in practice.

In this paper, we address *the satisfiability problem for Boolean combinations of convex non-linear constraints*. We follow the lazy SMT solving paradigm [7], where a classic David-Putnam-Logemann-Loveland (DPLL)-style SAT solving algorithm interacts with a theory solver based on fundamental results from convex programming. The theory solver needs only to check the feasibility of conjunctions of theory predicates passed onto it from the SAT solver. However, when all constraints are convex, a satisfying valuation can be found using interior point methods [8], running in polynomial time.

A central problem in a lazy SMT approach is for the theory solver to generate a compact explanation when the conjunction of theory predicates is unsatisfiable. We demonstrate how this can be achieved for convex constraints using duality theory for convex programming. Specifically, we formulate the convex programming problem in a manner that allows us to easily obtain the subset of constraints responsible for unsatisfiability.

Additionally, even when constraints are restricted to be convex, it is possible that, during Boolean reasoning, some of these constraints become negated, and thus the theory solver must handle some non-convex constraints. We show how to handle such constraints by set-theoretic reasoning and approximation with affine constraints.

The main novel contributions of our work can be summarized as follows:

- We present the first SMT solver for a Boolean combination of convex non-linear constraints. Our solver exploits information from the solution of convex optimization problems to establish satisfiability of conjunctions of convex constraints;
- We give a novel formulation that allows us to generate certificates of unsatisfiability in case the conjunction of theory predicates is infeasible, thus enabling the SMT solver to perform conflict-directed learning;
- Whenever non-convex constraints originate from convex constraints due to Boolean negation, we provide a procedure that can still use geometric properties of convex sets and supporting hyperplanes to generate approximations of the original set of constraints;
- We present a proof-of-concept implementation, CalCS, that can deal with a much broader category than linear arithmetic constraints, also including conic constraints, as the ones in quadratic and semidefinite programs, or any convex relaxations of other non-linear constraints [8]. We validate our approach on several benchmarks including formulas generated from BMC for hybrid systems and static analysis of floating-point programs, showing that our approach can be more accurate than current leading non-linear SMT solvers such as iSAT [9].

The rest of the paper is organized as follows. In Section II, we briefly review some related work in both areas on which this work is based, i.e. SMT solving for non-linear arithmetic constraints and convex optimization. In Section III, we describe background material including the syntax and semantics of the SMT problems our algorithm handles. Section IV introduces to the convex optimization concepts that our development builds on and provides a detailed explanation of our algorithm. In Section V we report implementation details on integrating convex and SAT solving. After presenting some benchmark results in Section VI, we conclude with a summary of our work and its planned extensions.

II. Related Work

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. Modern SAT and SMT solvers can efficiently find satisfying valuations of very large propositional formulae, including combinations of atoms from various decidable theories, such as lists, arrays, bit vectors [5]. However, extensions of the SMT problem to the theory of non-linear arithmetic constraints over the reals have only recently started to appear. Since our work combines both SAT/SMT solving techniques with convex programming, we briefly survey related works in both of these areas.

A. SMT solving for non-linear arithmetic constraints

Current SMT solvers for non-linear arithmetic adopt the lazy combination of a SAT solver with a theory solver for non-linear arithmetic. The ABsolver tool [10] adopts this approach to solve Boolean combinations of polynomial non-linear arithmetic constraints. The current implementation uses the numerical optimization tool IPOPT [11] for solving the non-linear constraints. However, without any other additional property for the constraints, such as convexity, the numerical optimization tool will necessarily produce incomplete results, and possibly incorrect, due to the local nature of the solver (all variables need upper and lower bounds). Moreover, in case of infeasibility, no rigorous procedure is specified to produce infeasibility proofs.

A completely different approach is adopted by the iSAT algorithm that builds on a unification of DPLL SAT-solving and interval constraint propagation [9] to solve arithmetic constraints. iSAT directly controls arithmetic constraint propagation from the SAT solver rather than delegating arithmetic decisions to a subordinate solver, and has shown superior efficiency. Moreover, it can address a larger class of formulae than polynomial constraints, admitting arbitrary smooth, possibly transcendental, functions. However, since interval consistency is a necessary, but not sufficient condition for real-valued satisfiability, spurious solutions can still be generated.

To reason about round-off errors in floating point arithmetic an efficient decision procedure (CORD) based on precise arithmetic and CORDIC algorithms has been recently proposed by Ganai and Ivancic [12]. In their approach, the non-linear part of the decision problem needs first to be translated into a linear arithmetic (LA) formula, and then an off-the-shelf SMT-LA solver and DPLL-style interval search are used to solve the linearized formula. For a given precision requirement, the approximation of the original problem is guaranteed to account for all inaccuracies.

B. Convex Programming

An SMT solver for the non-linear convex sub-theory is motivated by both theoretical and practical reasons. On the one hand, convex problems can be solved very efficiently today, and rely on a fairly complete and mature theory. On the other hand, convex problems arise in a broad variety of applications, ranging from automatic control systems, to communications, electronic circuit design, data analysis and modeling [8]. The solution methods have proved to be reliable enough to be embedded in computer-aided design or analysis tool, 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 a cheaply computable lower bounds via constraint or Lagrangian relaxations. A thorough reference on convex programming and its applications can be found in [8].

As an example, convex optimization has been used in electronic circuit design to solve the sizing problem [13]–[15]. Robust design approaches based on convex models of mixed-signal integrated circuits have also been presented in [16], [17]. While, in these cases, there was no Boolean structure, Boolean combinations of convex constraints arise when the circuit topology is not fixed, or for cyber-physical systems where continuous time dynamics need to be co-designed with discrete switching behaviors between modes. It is therefore necessary to have solvers that can reason about both Boolean and convex constraints.

In the context of optimal control design for hybrid systems, the work in [18], [19] proposes a combined approach of mixed-integer-programming (MIP) and constraint satisfaction problems (CSP), and specifically, convex programming and SAT solvers, as in our work. The approach in [18], [19] is, in some respects, complementary to ours. A SAT problem is first used to perform an initial logic inference and branching step on the Boolean constraints. Convex relaxations of the original MIP (including Boolean variables) are then solved by the optimization routine, which iteratively calls the SAT solver to ensure that the integer solution obtained for the relaxed problem is feasible and infer an assignment for the logic variables that were assigned to fractional values from the MIP. However, the emphasis in [18], [19] is more on speeding up the optimization over a set of mixed convex and integer constraints, rather than elaborating a decision procedure to verify feasibility of Boolean combinations of convex constraints, or generate infeasibility proofs. Additionally, unlike [18], [19], by leveraging conservative approximations, our work can also handle disjunctions of convex constraints.

III. Background and Terminology

We cover here some background material on convexity and define the syntax of the class of SMT formulae of our interest.

Convex Functions. A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is termed *convex* if its domain $\mathbf{dom} f$ is a convex set and if for all $x, y \in \mathbf{dom} f$, and θ with $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y). \qquad (1)$$

Geometrically, this inequality means that the *chord* from x to y lies above the graph of f. As a special case, when (1) always holds as an equality, then f is *affine*. All linear functions are also affine, hence convex. It is possible to recognize whether a function is convex based on certain properties. For instance, if f is differentiable, then f is convex if and only if $\mathbf{dom} f$ is convex and $f(y) \geq f(x) + \nabla f(x)^T (y - x)$ holds for all $x, y \in \mathbf{dom} f$, and $\nabla f(x)$ is the gradient of f. The above inequality states that if f is convex, its first-order Taylor approximation is always a global underestimator. The converse result can be also shown to be true. If f is twice differentiable, then f is convex if and only if $\mathbf{dom} f$ is convex and its Hessian $\nabla^2 f(x)$ is positive semidefinite matrix for all $x \in \mathbf{dom} f$. In addition to linear, affine, and positive semi-definite quadratic forms, examples of convex functions may include exponentials (e.g. e^{ax}), powers (e.g. x^a when $a \geq 1$), logarithms (e.g. $-\log(x)$), the $\max$ function, and all norms.

Convex Constraint. A convex constraint is of the form $f(x)\ \{<, \leq, >, \geq\}\ 0$ or $h(x) = 0$, where $f(x)$ and $h(x)$ are convex and affine (linear) functions, respectively, of their real variables $x \in \mathcal{D} \subseteq \mathbb{R}^n$, with $\mathcal{D}$ being a convex set. In the following, we also denote a constraint in the form $f(x) \leq 0$ ($f(x) < 0$) as a *convex* (*strictly convex*) constraint (CC), where $f(x)$ is a convex function on its convex domain. A convex constraint is associated with a set $\mathcal{C} = \{x \in \mathbb{R}^n : f(x) \leq 0\}$, i.e. the set of points in the space that satisfy the constraint. Since $\mathcal{C}$ is the 0-sublevel set of the convex function $f(x)$, $\mathcal{C}$ is also convex. We further denote the negation of a (strictly) convex constraint, expressed in the form $f(x) > 0$ ($f(x) \geq 0$), as *reversed convex* (*reversed strictly convex*) constraint (RCC). An RCC is, in general, non-convex as well as its satisfying set $\mathcal{N} = \{x \in \mathbb{R}^n : f(x) > 0\}$. The complement $\bar{\mathcal{N}}$ of $\mathcal{N}$ is, however, convex.

Syntax of Convex SMT Formulae. We represent SMT formulae over convex constraints to be quantifier-free formulae in conjunctive normal form, with atomic propositions ranging over propositional variables and arithmetic constraints. The formula syntax is therefore as follows:

$$\begin{array}{rcl} formula & ::= & \{clause \wedge\}^* clause \\ clause & ::= & (\{literal \vee\}^* literal) \\ literal & ::= & atom | \neg atom \\ atom & ::= & conv\_constraint \mid bool\_var \\ conv\_constraint & ::= & equation \mid inequality \\ equation & ::= & affine\_function = 0 \\ inequality & ::= & convex\_function \ relation \ 0 \\ relation & ::= & < \ \mid \ \leq \end{array}$$

In the grammar above, *bool_var* denotes a Boolean variable, and *affine_function* and *convex_function* denote affine and convex functions respectively. The terms *atom* and *literal* are used as is standard in the SMT literature. Note that the only theory atoms are convex or affine constraints. Even though we allow negations on convex constraints (hence allowing non-convex constraints), we will term the resulting SMT formula as a *convex SMT formula*.

Our constraint formulae are interpreted over valuations $\mu \in (BV \rightarrow \mathbb{B}) \times (RV \rightarrow \mathbb{R})$, where *BV* is the set of Boolean and *RV* the set of real-valued variables. The definition of satisfaction is also standard: a formula ϕ is satisfied by a valuation μ ($\mu \models \phi$) iff all its clauses are satisfied, that is, iff at least one atom is satisfied in any clause. A literal l is satisfied if $\mu_{\mathbb{B}}(l) =$`true`. Satisfaction of real constraints is with respect to the standard interpretation of the arithmetic operators and the ordering relations over the reals.

Based on the above definitions, here is an example of a convex SMT formula:

$$\begin{aligned} &(x + y - 3 = 0 \vee a \vee -\log(\mathrm{e}^x + \mathrm{e}^y) + 10 \geq 0) \\ &\wedge (\neg b \vee ||(x-2, z-3)||_2 \leq y - 5) \wedge (x^2 + y^2 - 4x \leq 0) \\ &\wedge \left(\neg a \vee y < 4.5 \vee \max\{2x + z, 3x^2 + 4y^4 - 4.8\} < 0\right), \end{aligned} \tag{2}$$

where $a, b \in BV$, $x, y, z \in RV$, and $|| \cdot ||_2$ is the Euclidean norm on $\mathbb{R}^2$.

If the SMT formula does not contain any negated convex constraint, the formula is termed a *monotone convex SMT formula*.

IV. Theory Solver for Convex Constraints

In optimization theory, the problem of determining whether a set (conjunction) of constraints are consistent, and if so, finding a point that satisfies them, is a *feasibility problem*. The feasibility problem for convex constraints can be expressed in the form

$$\begin{array}{rll} \text{find} & x & \\ \text{subject to} & f_i(x) \leq 0, & i = 1, \ldots, \ m \\ & h_j(x) = 0, & j = 1, \ldots, p \end{array} \tag{3}$$

where the single (vector) variable $x \in \mathbb{R}^n$ represents the n-tuple of all the real variables $(x_1, \ldots, x_n)^T$, the f_i functions are convex, and the h_j functions are affine. As in any optimization problem, if x is a feasible point and $f_i(x) = 0$, we say that the i-th inequality constraint $f_i(x) \leq 0$ is *active* at x. If $f_i(x) < 0$, we say the constraint $f_i(x) \leq 0$ is *inactive*. The equality constraints are active at all feasible points. For succinctness of presentation, we make the assumption that inequalities are non-strict (as listed in (3)), but our approach extends to systems with strict inequalities as well.

In this section, we describe how we construct a theory solver for a convex SMT formula that generates explanations when a system of constraints is infeasible. In general, the system of constraints can have both convex constraints and negated convex constraints (which can be non-convex). We will first consider the simpler case where all constraints in the system are convex, and show how explanations for infeasibility can be constructed by a suitable formulation that leverages duality theory (Section IV-A). We later give an alternative formulation (Section IV-B) and describe how to deal with the presence of negated convex constraints (Section IV-C).

Although it is possible to directly solve feasibility problems by turning them into optimization problems in which the objective function is identically zero [8], no information about the reasons for inconsistency would be propagated with this formulation, in case of infeasibility. Therefore, we cast the feasibility problem (3) as a combination of optimization problems with the addition of slack variables. Each of these newly generated problems is an equivalent formulation of the original problem (and it is therefore in itself a feasibility problem), while at the same time being richer in informative content. In particular, given a conjunction of convex constraints, our framework builds upon the following equivalent formulations of (3), namely the *sum-of-slacks* feasibility problem (*SSF*), and the *single-slack* feasibility (*SF*) problem, both detailed below.

A. Sum-of-Slacks Feasibility Problem

In the *SSF* problem, a slack variable s_i is introduced for every single constraint, so that (3) turns into the following

$$\begin{array}{rll} \text{minimize} & \sum_{k=1}^{m+2p} s_k & \\ \text{subject to} & \tilde{f}_k(x) - s_k \leq 0, & k = 1, \ldots, \ m + 2p \\ & s_k \geq 0 & \end{array} \tag{4}$$

where $\tilde{f}_k(x) = f_k(x)$ for $k = 1, \ldots, m$, $\tilde{f}_{m+j}(x) = h_j(x)$, and $\tilde{f}_{m+p+j}(x) = -h_j(x)$ for $j = 1, \ldots, p$. In other words, every equality constraint $h_j(x) = 0$ is turned into a conjunction of two inequalities, $h_j(x) \leq 0$ and $-h_j(x) \leq 0$ before applying the reduction in (4). The *SSF* problem can be interpreted as trying to minimize the infeasibilities of the constraints, by pushing each slack variable to be as much as possible close to zero. The optimum is zero and is achieved if and only if the original set of constraints (3) is feasible.

Based on *duality theory* [8], a *dual problem* is associated with (4), which maximizes the *Lagrange dual function* associated with (4), under constraints on the *dual variables* or *Lagrange multipliers*. While the dual optimal value always provides a lower bound to the original (*primal*) optimum, an important case obtains when this bound is tight and the two primal and dual optima coincide (*strong duality*). As a simple sufficient condition, Slater's theorem states that strong duality holds if the problem is convex, and there exists a strictly feasible point, such that the non-linear inequality constraints hold with strict inequalities. As a consequence of duality theory, the following result holds for (4) at optimum:

Proposition IV.1. *Let* $(x^*, s^*) \in \mathbb{R}^{n+m+2p}$ *be a primal optimal and* $z^* \in \mathbb{R}^{m+2p}$ *be a dual optimal point for* (4). *Then: (i) if* (3) *is feasible,* x^* *provides a satisfying assignment; (ii) moreover, we obtain:*

$$z_k^*(\tilde{f}_k(x^*) - s_k^*) = 0 \quad k = 1, \ldots, m+2p. \tag{5}$$

Proof sketch: The first statement trivially follows from the solution of problem (4). Since x^* is the optimal point, it also satisfies all the constraints in (4) with $s_k = s_k^* = 0$, therefore it is a satisfying assignment for (3). The second statement follows from *complementary slackness*. In fact, under the assumptions in Section III, (4) is a convex optimization problem. Moreover, it is always possible to find a feasible point which strictly satisfies all the nonlinear inequalities since, for a any given x, the slack variables s_k can be freely chosen, hence Slater's conditions hold. As a result, strong duality holds as well, i.e. both the primal and dual optimal values are attained and equal, which implies complementary slackness, as in (5). □

We use complementary slackness to generate infeasibility certificates for (3). In fact, if a constraint k is strictly satisfied (i.e. $s_k^* = 0$ and $\tilde{f}_k(x^*) < 0$) then the relative dual variable is zero, meaning that the constraint $\tilde{f}_k(x^*) \leq 0$ is actually non-active. Conversely, a non-zero dual variable will necessary correspond to either an unfeasible constraint ($s_k^* > 0$) or to a constraint that is non strictly satisfied ($s_k^* = 0$). In both cases, the constraint $\tilde{f}_k(x^*) \leq s_k$ is active at optimum and it is one of the reasons for the conflict. We can therefore conclude with the following result:

Proposition IV.2. *The subset of constraints in* (4) *that are related to positive dual variables at optimum represents the active subset, and therefore provides a succinct reason of infeasibility (certificate).* □

Numerical issues must be considered while implementing this approach. When (3) is feasible, the optimization algorithm in practice will terminate with $|\sum_{k=1}^{m+2p} s_k| \leq \epsilon_t$, thus producing an ϵ_t-suboptimal point for arbitrary small, positive ϵ_t. Accordingly, to enforce strict inequalities such as $\tilde{f}_k(x) < 0$, we modify the original expression with an additional user-defined positive slack constant ϵ_s as $\tilde{f}_k(x) + \epsilon_s \leq 0$, thus requiring that the constraint be satisfied with a desired margin ϵ_s. All the above conclusions valid for (3) can then be smoothly extended to the modified problem.

B. Single-Slack Feasibility Problem

While the *SSF* problem is the workhorse of our decision procedure, we also present an alternative formulation of the feasibility problem, which will be useful in the approximation of RCCs.

The *SF* problem minimizes the maximum infeasibility s of a set of convex constraints as follows

$$\begin{aligned} &\text{minimize} \quad s \\ &\text{subject to} \quad \tilde{f}_k(x) - s \leq 0, \quad k = 1, \ldots,\ m+2p \end{aligned} \tag{6}$$

where inequalities are pre-processed as in Section IV-A. The goal is clearly to drive the maximum infeasibility below zero. At optimum the sign of the optimal value s^* provides feasibility information. If $s^* < 0$, (6) has a strictly feasible solution; if $s^* > 0$ then (6) is infeasible; finally, if $s^* = 0$ (in practice $|s^*| \leq \epsilon_t$ for some small $\epsilon_t > 0$) and the minimum is attained, then the set of inequalities is feasible, but not strictly feasible. As in (4), complementary slackness will hold at optimum, i.e.

$$z_k^*(\tilde{f}_k(x^*) - s^*) = 0 \quad k = 1, \ldots, m+2p.$$

Therefore, even when the problem is feasible, whenever a constraint k is not active, then $(\tilde{f}_k(x^*) - s^*) \neq 0$ will be strictly satisfied, and imply $z_k = 0$. Conversely, if $z_k \neq 0$, then the constraint $(\tilde{f}_k(x^*) - s^*)$ is certainly active and $\tilde{f}_k(x)$ contributes to determine the maximum infeasibility for the given problem, in the sense that if s^* was further pushed to be more negative, $\tilde{f}_k(x)$ would be no longer satisfied.

C. Dealing with Reversed Convex Constraint

A negated (reversed) convex constraint (an RCC) is non-convex and defines a non-convex set $\mathcal{N}$. Any conjunction of these non-convex constraints with other convex constraints results in general in a non-convex set. To deal with such non-convex sets, we propose heuristics to compute convex over- and under-approximations, which can then be solved efficiently. This section describes these techniques.

Our approximation schemes are based on noting that the complementary set $\bar{\mathcal{N}}$ is convex. Therefore geometric properties of convex sets, such as strict or weak separation [8], can still be used to approximate or bound $\mathcal{N}$ via a supporting hyperplane. Once a non-convex constraint is replaced with a bounding hyperplane, the resulting *approximate problem* (AP) will again be convex, and all the results in Section IV-A will be valid for this approximate problem.

For simplicity, we assume in this section that we have exactly one non-convex constraint (RCC), and the rest of the constraints are convex. We will describe the general case in Sec. IV-D. Let $g(x)$ be the convex function associated with the RCC. Our approach proceeds as follows:

1) Solve the sum-of-slacks (SSF) problem for just the convex constraints. Denote the resulting convex region by $\mathcal{B}$.
   If the resulting problem is `UNSAT`, report this answer along with the certificate computed as described in Sec. IV-A.
   Otherwise, if the answer returned is `SAT`, denote the optimal point as x_b^* (satisfying assignment) and proceed to the next step.

2) Add the negation of the RCC (a convex constraint) and solve the SSF problem again, which we now denote as *reversed problem* (RP). There are two cases:
   (a) If the answer is UNSAT, then the RCC region $\bar{\mathcal{N}}$ does not intersect the convex region $\mathcal{B}$. This implies that $\mathcal{B} \subset \mathcal{N}$, and hence the RCC is a redundant constraint. This situation is illustrated in Fig. 1(a). Thus, the solver can simply return SAT (as returned in the previous step).
   (b) On the other hand, if the answer is SAT, we denote as x_c^* the optimal point of the RP and check whether the negated RCC is now redundant, based on the shift induced in the optimal point x_b^*. In particular, if both x_c^* and x_b^* are inside $\bar{\mathcal{N}}$, we solve two single-slack feasibility (*SF*) problems, and we denote as $\tilde{x}_b^*$ and $\tilde{x}_c^*$ the two optimal points, for the problem having just the convex constraints and for the the RP, respectively. Similarly, we denote the two optimal values as $\tilde{s}_b^*$ and $\tilde{s}_c^*$.
   As also observed in Section IV-B, for a set of satisfiable constraints, $\tilde{x}_b^*$, $\tilde{x}_c^*$, $\tilde{s}_b^*$ and $\tilde{s}_c^*$ may contain more information than the optimal points x_b^* and x_c^* (and their slack variables) for the *SSF* problem. In fact, since $\tilde{s}_b^*$ and $\tilde{s}_c^*$ are also allowed to assume negative (hence different) values at optimum, they can provide useful indications on how the RCC has changed the geometry of the feasible set, and which constraints are actually part of its boundary, thus better driving our approximation scheme. In particular, if we verify that $\tilde{s}_b^* = \tilde{s}_c^*$, $\tilde{x}_b^* = \tilde{x}_c^*$, and $\mathcal{B} \subset \bar{\mathcal{N}}$, then we imply $\mathcal{B} \cap \mathcal{N} = \emptyset$. Hence, the solver can return UNSAT. Techniques to detect if a conjunction of convex constraints generates sets that are (exactly or approximately) contained in a convex set are reported in [20], [21]. For instance, when both $\mathcal{B}$ and $\bar{\mathcal{N}}$ are spheres, the condition $\mathcal{B} \subset \bar{\mathcal{N}}$ is equivalent to checking that the slack constraint related to the RCC is not active at optimum in the *SF* problem. This case is illustrated in Fig. 1(b) for the following conjunction of constraints:

$$(x_1^2 + x_2^2 - 1 \leq 0) \wedge (x_1^2 + x_2^2 - 4 > 0)$$

   where $(x_1^2+x_2^2-4 > 0)$ is the non-convex constraint defining region $\mathcal{N}$. If set containment cannot be exactly determined the procedure returns UNKNOWN.

   If none of the above cases hold, we proceed to the next step. For example, this is the case whenever x_b^* is outside $\bar{\mathcal{N}}$, or on its boundary (i.e. $g(x_b^*) \geq 0$). This implies that the negated RCC is not redundant, and we can move to the next step without solving the two *SF* problems.
3) In this step, we generate a convex under-approximation of the original formula including the convex constraints and the single non-convex RCC. If the resulting problem is found satisfiable, the procedure returns SAT. Otherwise, it returns UNKNOWN.

We now detail the under-approximation procedure in Step 3. As an illustrative example, we use a 2-dimensional region defined by the following SMT formula:

$$(x_1^2+x_2^2-1 \leq 0)\wedge(x_1^2+x_2^2-4x_1 \leq 0)\wedge(x_1^2+x_2^2-2x_2 > 0). \tag{7}$$

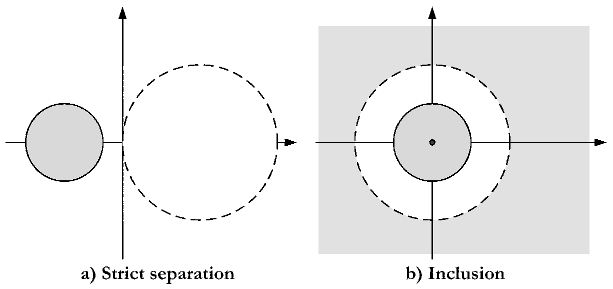

Fig. 1. Two special cases for handling non-convex constraints: (a) by adding a negated RCC a new set is generated that is strictly separated from the previous convex set; (b) the negated RCC generates a set that totally includes the previous convex set.

As apparent from the geometrical representation of the sets in Fig. 2 a), the problem is clearly satisfiable and a satisfying valuation could be any point in the grey region $\mathcal{A}$.

First, we note for this example the results obtained before the under-approximation is performed. We solve the *SSF* problem for the convex set $\mathcal{B} = \{(x_1, x_2) \in \mathbb{R}^2 : (x_1^2 + x_2^2 - 1 \leq 0) \wedge (x_1^2 + x_2^2 - 4x_1 \leq 0)\}$, obtained from $\mathcal{A}$ after dropping the RCC N. The problem is feasible, as shown in Fig. 2 (b), and the optimal point $x_b^* = (0.537, 0)$ is returned.

Next, the RCC is negated to become convex and the *SSF* problem is now solved on the newly generated formula

$$(x_1^2+x_2^2-1 \leq 0)\wedge(x_1^2+x_2^2-4x_1 \leq 0)\wedge(x_1^2+x_2^2-2x_2 \leq 0)$$

which represents the previously defined (RP). The RP will provide useful information for the approximation, thus acting as a "geometric probe" for the optimization and search space. Since the RCC is reversed, the RP is convex and generates the set $\mathcal{C}$, shown in Fig. 2 (c).

Let us assume, at this point, that the RP is feasible, as in this example. Then $\mathcal{C} \neq \emptyset$, and an optimal point $x_c^* = (0.403, 0.429) \in \mathcal{C}$ is provided. Moreover, $\mathcal{A}$ can be expressed as $\mathcal{B} \setminus \mathcal{C}$, and x_b^* is clearly outside the convex set $\bar{\mathcal{N}}$ generated by the negated RCC, meaning that we can go to the under-approximation step without solving the SF problems since the negated RCC is certainly non-redundant.

The key idea for under-approximation is to compute a hyperplane that we can use to separate the RCC region $\mathcal{N}$ from the remaining convex region. This "cut" in the feasible region is performed by exploiting the perturbation of the optimal point from x_b^* to x_c^* induced by the negated RCC $\bar{N} : (x_1^2 + x_2^2 - 2x_2) \leq 0$. At this point, we examine a few possible cases:

Case (i): Suppose that $x_b^* \neq x_c^*$, and x_b^* is outside $\bar{\mathcal{N}}$ (as in our example). In this case, we find the orthogonal projection $p = \mathcal{P}(x_b^*)$ onto $\bar{\mathcal{N}}$, which can be performed by solving a convex, L_2-norm minimization problem [8]. Intuitively, this corresponds to projecting x_b^* onto a point p on the boundary of the region $\bar{\mathcal{N}}$. Finally, we compute the supporting hyperplane to $\bar{\mathcal{N}}$ in p. The half-space defined by this hyperplane that excludes $\bar{\mathcal{N}}$ provides our convex (affine) approximation $\tilde{\mathcal{N}}$ for $\mathcal{N}$.

For our example, $\bar{\mathcal{N}} = \{x \in \mathbb{R}^n : x_1^2 + x_2^2 - 2x_2 \leq 0\}$. The affine constraint resulting from the above procedure is $\tilde{N} : -0.06x_1 + 0.12x_2 + 0.016 < 0$. On replacing the RCC N with $\tilde{N}$, we obtain a new set $\mathcal{D}$, as shown Fig. 2(d), which is now our approximation for $\mathcal{A}$.

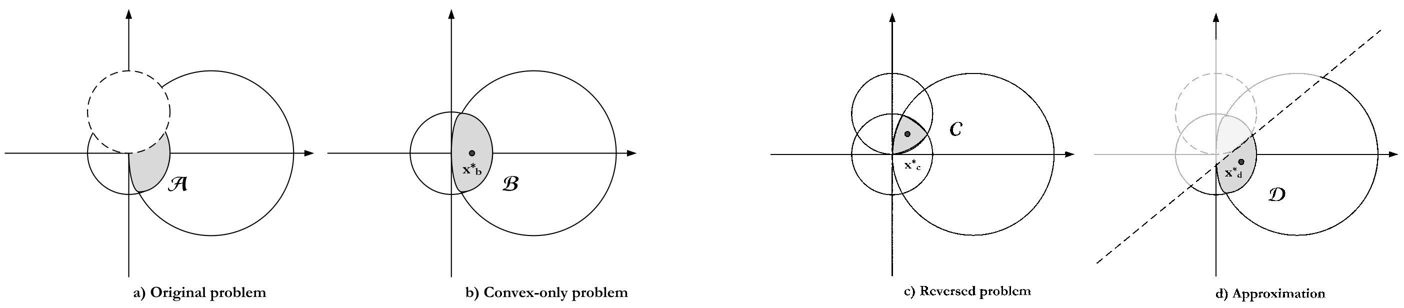

Fig. 2. Geometrical representation of the sets used in Section IV-C to illustrate the approximation scheme in CalCS: (a) $\mathcal{A}$ is the search space (in grey) for the original non-convex problem including one RCC constraint; (b) $\mathcal{B}$ is search space when the RCC is dropped (over-approximation of $\mathcal{A}$); (c) $\mathcal{C}$ is the search space for the *reversed problem*, i.e. the problem obtained from the original one in (a) when the RCC is negated; the RP is therefore convex; (d) $\mathcal{D}$ is the under-approximation of $\mathcal{D}$ in (a) using a supporting hyperplane.

An SSF problem can now be formulated for $\mathcal{D}$ thus providing the satisfying assignment $x^*_d = (0.6, -0.33)$. The approximation procedure will stop here and return SAT.

Notice that, whenever x^*_b is on the boundary of $\bar{\mathcal{N}}$, a similar approximation as described above can be performed. In this case, x^*_b is the point through which the supporting hyperplane needs to be computed, and no orthogonal projection is necessary. The normal direction to the plane needs, however, to be numerically computed by approximating the gradient of $g(x)$ in x^*_b.

Case (ii): A second case occurs when $x^*_b \neq x^*_c$, but both x^*_b and x^*_c are inside $\bar{\mathcal{N}}$. In this case, starting from x^*_c we search the closest boundary point along the $(x^*_b - x^*_c)$ direction, and then compute the supporting hyperplane through this point as in the previous case. In fact, to find an under-approximation for the feasible region $\mathcal{A}$, we are looking for an *over-approximation* of the set $\bar{\mathcal{N}}$ in the form of a tangent hyperplane. Since the optimal point x^*_b moves to x^*_c after the addition of the negated RCC, $\bar{\mathcal{N}}$ will be more "centered" around x^*_c than around x^*_b. Therefore, a reasonable heuristic could be to pick the direction starting from x^*_c and looking outwards, namely $(x^*_b - x^*_c)$.

Case (iii): Assume now that $x^*_b = x^*_c$ (with both x^*_b and x^*_c inside $\bar{\mathcal{N}}$), but we have $\tilde{x}^*_b \neq \tilde{x}^*_c$, where $\tilde{x}^*_b$ and $\tilde{x}^*_c$ are the two optimal points, respectively, for the *SF* problem having just the convex constraints and for the the RP in the *SF* form, as computed in Step 2 (b) above. In this case, to operate the "cut", we cannot use the perturbation on x^*_b and x^*_c, as in Case (ii), but we can still exploit the information contained in the SF problems. This time, starting from $\tilde{x}^*_c$, we search the closest boundary point along the $(\tilde{x}^*_b - \tilde{x}^*_c)$ direction, and then compute the supporting hyperplane through this boundary point.

Case (iv): Finally, both $x^*_b = x^*_c$ and $\tilde{x}^*_b = \tilde{x}^*_c$ can also occur, as for the following formula:

$$(x_1^2 + x_2^2 - 1 \geq 0) \wedge (x_1^2 + x_2^2 - 4 \leq 0),$$

for which $\mathcal{A}$ would coincide with the white ring region in Fig. 1 b) (including the dashed boundary). In this case, no useful information can be extracted from perturbations in the optimal points. The feasible set appears "isotropic" to both x^*_b and $\tilde{x}^*_b$, meaning that any direction could potentially be chosen for the approximations. In our example, we infer from the *SF* problems that the inner circle is the active constraint and we need to replace the non-convex constraint corresponding to its exterior with a supporting hyperplane, e.g. $-x_1 + 1 \leq 0$, by simply picking it to be orthogonal to one of the symmetry axes of the feasible set. The resulting under-approximation is found SAT and we obtain a satisfying assignment consistent with this approximation.

This completes the description of the under-approximation procedure of Step 3. We note that we still have the possibility for the solver to return UNKNOWN. Depending on the target application, the user can interpret this as SAT (possibly leading to spurious counterexamples in BMC) or UNSAT (possibly missing counterexamples). For higher accuracies, the approximation scheme can also be iterated over a set of boundary points of the original constraint $f(x)$, to build a finer polytope bounding the non-convex set.

D. Overall Algorithm

Our theory solver is summarized in Fig. 3. This procedure generalizes that described in the preceding section by handling multiple reversed convex constraints (RCCs). In essence, if the conjunction of all convex constraints and any single RCC is found UNSAT, then we report UNSAT. In order to report SAT, on the other hand, we must consider all convex constraints and all affine under-approximations of non-convex constraints.

The details are as follows. For a given conjunction of CCs and RCCs, we first solve the *SSF* problem generated by the CCs alone (Section IV-A). If the problem is UNSAT, the algorithm returns the subset of constraints that provide the reason for inconsistency (infeasibility certificate) and stops. Otherwise, each RCC is processed sequentially. For each RCC, the initial convex problem is augmented and the RP is formulated and solved. If the RP is unfeasible then, as discussed in Section IV-C, the constraint is ignored since it is non-active for the current feasibility problem. On the contrary, if the RP is feasible we proceed by computing an approximation.

The `Approximate` method implements the under-approximation strategies outlined in Section IV-C and determines whether the constraint is non-active or can be dropped by solving additional SF problems (Section IV-B). If the negated RCC is fully included in the set generated by the CCs alone, (e.g. the megated RCC and the set generated by the CCs are both circles and the negated RCC is non-active for the RP) the problem is UNSAT, meaning that the RCC is incompatible with the whole set of CC (step 2(b) of Section IV-C). The full set, including both the CC and the

```
function [status, out] = Decision_Manager(CC, RCC)
% receive a set of convex (CC) and non-convex constraints (RCC)
% return SAT/UNSAT/UNKNOWN and MODEL/CERTIFICATE
%
% solve sum-of-slacks feasibility problems with CCs
[status, out] = SoS_solve(CC);
% OUT contains CERTIFICATE
if (status == UNSAT) return; end
AC = CC;% AC stores all constraints
for (k = 1, k <= length(RCC), k++)
    RP = reverse(CC, RCC(k));
    [status, out] = SoS_solve(RP);
    % strict separation: ignore RCC
    if (RP == UNSAT) continue; end
    % both CC and RP problems are SAT: approximation
    [approxCC, active, drop] = Approximate(RCC(k));
    % RCC incompatible (inclusion)
    if (˜active)
        status = UNSAT;
        % certificate
        out = [CC, RCC(k)]; return;
        % over-approximation: ignore constraint
    elseif (drop) continue;
    else AC = AC ∪ approxCC;
        [status, out] = SoS_solve(AC);
        if (status == SAT)
          Check SAT assignment on original constraints;
          if (original constraints satisfied) status = SAT; return;
          end
        end
    end
end
status = UNKNOWN;
```

Fig. 3. Pseudo-code for the CalCS decision procedure.

current RCC is returned as an explanation for the conflict. If an over-approximation is required, then the constraint is ignored. If the constraint is compatible and cannot be dropped, the supporting hyperplane is computed and the new under-approximated problem is solved. The algorithm proceeds with visiting the other RCCs. Finally, when all non-convex constraints have been processed without returning UNSAT the algorithm is re-invoked on the set of convex constraints CC and the set of affine under-approximations of non-convex constraints RCC. If this invocation returns SAT, so does the overall algorithm; otherwise, it returns UNKNOWN. A SAT answer is accompanied by a satisfying valuation to variables.

V. Integrating Convex Solving and SAT Solving

Using the theory solver described in Section IV, we have implemented a proof-of-concept SMT solver, CalCS, that supports the convex sub-theory. As in [10], CalCS receives as input an SMT formula in a DIMACS-like CNF format, where atomic predicates can be both Boolean or convex constraints, according to the definitions in Section III. Following the lazy theorem proving paradigm, the SMT problem is first transformed into a SAT problem, by mapping the nonlinear constraints into auxiliary Boolean variables. This Boolean abstraction of the original formula is then passed to the SAT solver. If the outcome is UNSAT, the theory manager terminates and returns UNSAT. Conversely, the assigned auxiliary variables are mapped back to a conjunction of CC and RCC and are sent to the theory for consistency checking. If the theory solver returns SAT, a combined Boolean and real *model* (satisfying assignment) is also returned. Otherwise, whenever inconsistencies are found (UNSAT), the reason for the conflict (certificate) is encoded into the *learned clause* $(\neg l_1 \vee \ldots \vee \neg l_k)$, $l_1, \ldots, l_k$ being the auxiliary literals associated with the infeasible constraints. The SAT problem is then augmented and new SAT queries are performed until either the SAT solver terminates with UNSAT or the theory solver with SAT. To benefit from the most recent advances in SAT solving, MiniSAT2 [22] is adapted to our requirements by adding decision heuristics to prune our search space. To reduce the number of theory calls, we first assign values to the Boolean variables so as to satisfy as many clauses as possible. Subsequently, we start assigning values to some of the auxiliary variables, until all clauses are satisfied. Whenever we need to decide an assignment for an auxiliary variable, we affirm any CC and negate any RCC as a first choice, to maximize the number of CCs for each theory call, hence the chances of deciding without approximations. The following theorems state the properties of CalCS.

Theorem V.1. *Let ϕ be a convex SMT formula. Then, if CalCS reports* SAT *on ϕ, ϕ is satisfiable. Alternatively, if CalCS reports* UNSAT*, ϕ is unsatisfiable.* □

Note that the converse does not hold in general. If CalCS reports UNKNOWN, it is possible that the formula ϕ is either satisfiable or unsatisfiable. In the case of a monotone convex SMT formula, we have stronger guarantees.

Theorem V.2. *Let ϕ^+ be a monotone convex SMT formula. Then, CalCS reports* SAT *on ϕ^+ iff ϕ^+ is satisfiable and CalCS reports* UNSAT *iff ϕ^+ is unsatisfiable.* □

The above result follows straightforwardly from the fact that for monotone convex SMT formulas, all convex constraints are assigned true, so the theory solver never sees non-convex constraints.

VI. Experimental Results

In our prototype implementation, we use the Matlab-based convex programming package CVX [23] to solve the optimization problems, while theory solver and SAT solver interact via an external file I/O interface. We therefore allow for all functions and operations supported by disciplined convex programming [24]. We first validated our approach on a set of benchmarks [25], including geometric decision problems dealing with the intersection of n-dimensional geometric objects, and randomly generated formulae obtained from 3-SAT classical Boolean benchmarks [26], after replacing some of the Boolean variables with convex or RC constraints. Table I shows a summary of an experimental evaluation of our tool, also in comparison with iSAT. To evaluate the impact of generating a compact explanation of unsatisfiability (a certificate) we run CalCS in two modes: in the first mode (C in Table I), a subset of conflicting constraints is provided, as detailed in Section IV, while in the second mode (NC in Table I), the full set of constraints is returned as simply being inconsistent. All benchmarks were performed on a 3 GHz Intel Xeon machine with 2 GByte physical memory running Linux.

Results show that whenever problems are purely convex, they are solved without approximation and with full control of rounding errors and can provide results that are more accurate than the ones of iSAT, in comparable time, in spite of our

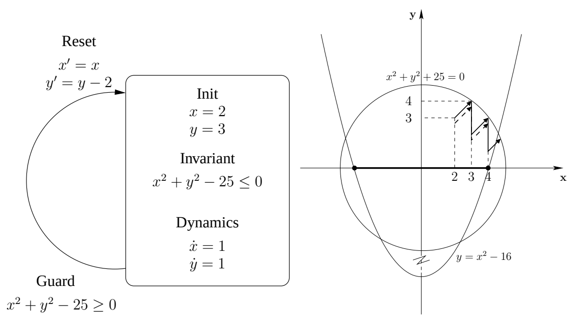

Fig. 4. Simple hybrid automata with convex guards and invariants (left) and representation of the error traces from CalCS (solid) and iSAT (dashed) in the (x, y) plane (right). The safety interval for x is $[-4, 4]$.

prototype implementation. In particular, the interval-based reasoning scheme can incur inaccuracies and large computation times when the satisfying sets are lower dimensional sets with respect to the full search space including all the real variables in the problems. As a simple example, for the formula:

$$(x_1^2 + x_2^2 - 1 \leq 0) \wedge (x_1^2 + x_2^2 - 6x_1 + 5 < 0), \qquad (8)$$

iSAT returns an interval that contains a spurious solution, while our convex sub-theory can rigorously deal with tight inequalities and correctly returns UNSAT (see (8) and Conj3 in Tab. I). Similarly, CalCS can provide the correct answer for the following formulae ((9) and (10) in Tab. I), mentioned as prone to unsound or spurious results in [12]:

$$(x+y < a) \wedge (x-y < b) \wedge (2x > a+b) \wedge (a = 1) \wedge (b = 0.1), \qquad (9)$$

$$(x \leq 10^9) \wedge (x + p > 10^9) \wedge (p = 10^{-8}). \qquad (10)$$

While for small problem instances (Bool1-2-3, Conj1) both the C and NC schemes show similar performances, the advantages of providing succinct certificates becomes evident for larger instances (Bool4-5-6-7, Conj2), where we rapidly reached a time-over (TO) limit (set to 200 queries to the theory solver) without certificates. A faster implementation would be possible by using commercial, or more optimized, convex optimization engines.

We have also tested CalCS on BMC problems, consisting in proving a property of a hybrid discrete-continuous dynamic system for a fixed unwinding depth k. We generated a set of hybrid automata (HA) including convex constraints in both their guards and invariants. For the simple HA in Fig. 4 we also report a pictorial view of the safety region for the x variable, and the error traces produced by CalCS (solid line) and iSAT (dashed line). The circle in Fig. 4 represents the HA invariant set, while the portion of the parabola underlying the x axis determines the set of points x satisfying the property we want to verify, i.e. $\{x \in \mathbb{R} : x^2 - 16 \leq 0\}$. Our safety region is therefore the closed interval $[-4, 4]$. The dynamics of the HA are represented by the solid and dash lines. As far as the invariant is satisfied, the continuous dynamics hold and the HA moves along the arrows on the (x, y) plane, starting from the point $(2, 3)$. When the trajectories intersect the circle's boundary, a jump occurs (e.g. from $(3, 4)$ to $(3, 2)$ and from $(4, 3)$ to $(4, 1)$) and the system is reset. Initially, both the solid and dashed trajectories are overlapped (they are drawn slightly apart for clarity). However, more accurately, we return unsafe after 3 BMC steps ($k = 3$), while iSAT stops at the second step producing an error trace that is still in the safety region, albeit on the edge. As an additional case study, we considered aircraft conflict resolution [27] based on the Air Traffic Alert and Collision Avoidance System (TCAS) specifications (Tab. II). The hybrid automata in Fig. 5 models a standardized maneuver that two airplanes need to follow when they come close to each other during their flight. When the airplanes are closer than a distance d_{near}, they both turn left by $\Delta\phi$ degrees (which is kept fixed to a constant value in our maneuver) and fly for a distance d along the new direction. Then they turn right and fly until their distance exceeds a threshold d_{far}. At this point, the conflict is solved and the two airplanes can return on their original route. We verified that the two airplanes stay always apart, even without coordinating their maneuver with the help of a central unit.

TABLE I
CalCS Experiments: In mode C the UNSAT core is provided while in mode NC the full set of constraints is returned as conflicting; Approx denotes the number of RCCs approximated as hyperplanes; S stands for SAT, U for UNSAT.

File	Res.	CalCS C/NC [s]	Approx C/NC	Queries C/NC	iSAT [s]
(8)	U	0.5 (U)	0	1	0.05 (S)
(9)	U	0.2 (U)	0	1	0 (S)
Conj3	U	$22/23$ (U)	5	3	0.05 (S)
(10)	S	0.2 (S)	0	1	0 (U)
Bool1	S	3.5 (S)	1	1	8 (S)
Bool2	S	16 (S)	3	1	0.91 (S)
Bool3	S	$27/23$ (S)	$5/4$	2	0.76 (S)
Conj1	U	$8.7/9.5$ (U)	3	2	0.3 (U)
Bool4	S	$17.9/17.7$ (S)	3	1	0.75 (S)
Conj2	U	$17/23.3$ (U)	$4/5$	$4/7$	0.4 (U)
Bool5	U	$23.5/321.7$ (U)	$4/36$	$5/94$	0.02 (U)
Bool6	U	$29.8/TO$ (U)	$5/-$	$6/-$	0.4 (U)
Bool7	S	$257.7/TO$ (S)	$24/-$	$6/-$	1.31 (S)

TABLE II
TCAS BMC Case Study

Maneuver type	Crash state	#queries	run time [s]
UNSAFE	CRUISE	2	10.9
UNSAFE	LEFT	4	28
UNSAFE	STRAIGHT	6	50
SAFE	NONE	10	110

Finally, we have applied CalCS to formulae generated in the context of static analysis of floating-point numerical code, and requiring an SMT solver that can handle non-linear arithmetic constraints over the reals. Tab. III summarizes the performance of CalCS on a set of benchmarks provided by Vo *et al.*, who are developing a static analyzer to detect floating-point exceptions (e.g., overflow and underflow) [28]. Early experience with CalCS on this set of benchmarks, mostly including conjunctions of linear and non-linear constraints, seems promising. After a fast pre-processing step, CalCS can deal with the formulae of interest providing an exact answer in reasonable computation time even when approximations are needed, which demonstrates that our solver can be general enough to be suitable for different application domains.

VII. Conclusions

We have proposed a procedure for satisfiability solving of a Boolean combination of non-linear constraints that are convex. Our prototype SMT solver, CalCS, combines fundamental

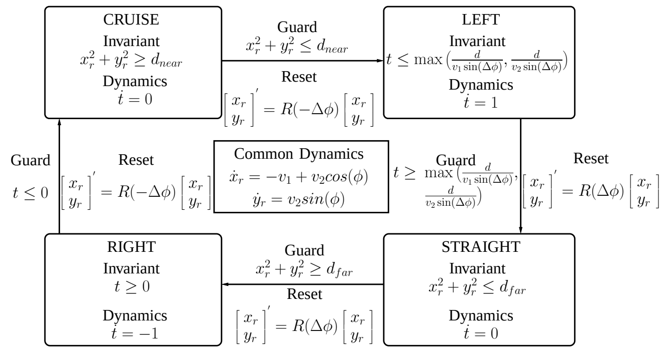
CRUISE
Invariant
$x_r^2 + y_r^2 \geq d_{near}$
Dynamics
$\dot{t} = 0$
Guard
$x_r^2 + y_r^2 \leq d_{near}$
Reset
$\begin{bmatrix} x_r \\ y_r \end{bmatrix}' = R(-\Delta\phi) \begin{bmatrix} x_r \\ y_r \end{bmatrix}$
LEFT
Invariant
$t \leq \max\left(\frac{d}{v_1 \sin(\Delta\phi)}, \frac{d}{v_2 \sin(\Delta\phi)}\right)$
Dynamics
$\dot{t} = 1$
Guard
$t \leq 0$
Reset
$\begin{bmatrix} x_r \\ y_r \end{bmatrix}' = R(-\Delta\phi) \begin{bmatrix} x_r \\ y_r \end{bmatrix}$
Common Dynamics
$\dot{x}_r = -v_1 + v_2 cos(\phi)$
$\dot{y}_r = v_2 sin(\phi)$
Guard
$t \geq \max\left(\frac{d}{v_1 \sin(\Delta\phi)}, \frac{d}{v_2 \sin(\Delta\phi)}\right)$
Reset
$\begin{bmatrix} x_r \\ y_r \end{bmatrix}' = R(\Delta\phi) \begin{bmatrix} x_r \\ y_r \end{bmatrix}$
RIGHT
Invariant
$t \geq 0$
Dynamics
$\dot{t} = -1$
Guard
$x_r^2 + y_r^2 \geq d_{far}$
Reset
$\begin{bmatrix} x_r \\ y_r \end{bmatrix}' = R(\Delta\phi) \begin{bmatrix} x_r \\ y_r \end{bmatrix}$
STRAIGHT
Invariant
$x_r^2 + y_r^2 \leq d_{far}$
Dynamics
$\dot{t} = 0$

Fig. 5. Air Traffic Alert and Collision Avoidance System

TABLE III
BENCHMARKS FROM STATIC ANALYSIS OF NUMERICAL CODE: APPROX DENOTES THE NUMBER OF RCCS APPROXIMATED AS HYPERPLANES.

File	Result	Time [s]	Approx
Num1	SAT	1.08	0
Num2	SAT	4.35	2
Num3	UNSAT	0.55	0
Num4	UNSAT	0.55	0
Num5	SAT	4.27	2
Num6	SAT	0.49	0
Num7	SAT	2.82	1
Num8	UNSAT	2.64	2
Num9	UNSAT	2.10	0
Num10	UNSAT	0.53	0
Num11 $-$ 13	UNSAT	0	0
Num14	UNSAT	1.91	0
Num15	UNSAT	1.94	0
Num16	UNSAT	0.53	0
Num17 $-$ 18	UNSAT	0	0
Num19	UNSAT	0.49	0
Num20	UNSAT	0	0

results from convex programming with the efficiency of SAT solving. By restricting our domain to a subset of non-linear constraints, we can solve for conjunctions of constraints globally and accurately, by formulating a combination of convex optimization problems and exploiting information from their primal and dual optimal values. When the conjunction of theory predicates is infeasible, our formulation can generate certificates of unsatisfiability, thus enabling conflict-directed learning. Finally, whenever non-convex constraints originate from convex constraints due to Boolean negation, our procedure uses geometric properties of convex sets to generate conservative approximations of the original set of constraints. Experiments on several benchmarks, including examples of BMC for hybrid systems, show that CalCS can be more accurate than other state-of-the-art non-linear SMT solvers. In the future, we plan to further refine the proposed algorithms by devising more sophisticated learning and approximation schemes as well as more efficient implementations.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Gigascale Systems Research Center and the Multiscale Systems Center, two of six research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program. The third author is also supported in part by the NSF grant CNS-0644436, and by an Alfred P. Sloan Research Fellowship. Comments of Susmit Jha on drafts of this paper are also acknowledged.

REFERENCES

[1] D. Walter, S. Little, and C. Myers, “Bounded model checking of analog and mixed-signal circuits using an SMT solver,” in *Automated Technology for Verification and Analysis*, 2007, pp. 66–81.
[2] C. Tomlin, I. Mitchell, and R. Ghosh, “Safety verification of conflict resolution maneuvers,” *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 2, pp. 110–120, Jun. 2001.
[3] M. Franzle and C. Herde, “HySAT: An efficient proof engine for bounded model checking of hybrid systems,” *Formal Methods in System Design*, pp. 179–198, 2007.
[4] S. Jha, B. Brady, and S. A. Seshia, “Symbolic reachability analysis of lazy linear hybrid automata,” in *5th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, October 2007, pp. 241–256.
[5] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, *Satisfiability Modulo Theories, Chapter in Handbook of Satisfiability*. IOS Press, 2009.
[6] S. Ratschan, “Efficient solving of quantified inequality constraints over the real numbers,” *ACM Trans. Comput. Logic*, vol. 7, no. 4, pp. 723–748, 2006.
[7] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, “Solving SAT and SAT Modulo Theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T),” *Journal of the ACM*, vol. 53, no. 6, pp. 937–977, 2006.
[8] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univesity Press, 2004.
[9] M. Franzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige, “Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure,” in *JSAT Special Issue on SAT/CP Integration*, 2007, pp. 209–236.
[10] A. Bauer, M. Pister, and M. Tautschnig, “Tool-support for the analysis of hybrid systems and models,” in *Proc. of DATE*, 2007.
[11] “IPOPT,” https://projects.coin-or.org/Ipopt.
[12] M. Ganai and F. Ivancic, “Efficient decision procedure for non-linear arithmetic constraints using CORDIC,” in *Formal Methods in Computer-Aided Design, 2009. FMCAD 2009*, 15-18 2009, pp. 61–68.
[13] J. P. Fishburn and A. E. Dunlop, “TILOS: A posynomial programming approach to transistor sizing,” in *IEEE International Conference on Computer- Aided Design: ICCAD-85*, 15-18 1985, pp. 326–328.
[14] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S.-M. Kang, “An exact solution to the transistor sizing problem for CMOS circuits using convex optimization,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 11, pp. 1621–1634, 1993.
[15] M. del Mar Hershenson, S. P. Boyd, and T. H. Lee, “Optimal design of a CMOS op-amp via geometric programming,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 1–21, 2001.
[16] X. Li, P. Gopalakrishnan, Y. Xu, and L. Pileggi, “Robust Analog/RF Design with Projection-Based Posynomial Modeling,” in *Proc. IEEE/ACM International Conference on CAD*, 2004, pp. 855–862.
[17] Y. Xu, K. Hsiung, X. Li, I. Nuasieda, S. Boyd, and L. Pileggi, “OPERA: OPtimization with Ellispoidal uncertainty for Robust Analog IC design,” 2005, pp. 632–637.
[18] A. Bemporad and N. Giorgetti, “A SAT-based hybrid solver for optimal control of hybrid systems,” in *Hybrid Systems: Computation and Control*, 2004.
[19] ——, “Logic-based solution methods for optimal control of hybrid systems,” *IEEE Transactions on Automatic Control*, vol. 51, no. 6, pp. 963–976, Jun. 2006.
[20] O. L. Mangasarian, “Set containment characterization,” *J. of Global Optimization*, vol. 24, pp. 473–480, 2002.
[21] V. Jeyakumar, “Characterizing set containments involving infinite convex constraints and reverse-convex constraints,” *SIAM J. Optimization*, vol. 13, pp. 947–959, 2003.
[22] “MiniSAT SAT solver,” http://minisat.se.
[23] M. Grant and S. Boyd, “CVX: Matlab software for disciplined convex programming, version 1.21,” http://cvxr.com/cvx, May 2010.
[24] M. Grant, S. Boyd, and Y. Ye, “Disciplined convex programming,” in *Global Optimization: From Theory to Implementation*, 2006, pp. 155–210.
[25] “Benchmarks,” http://www.eecs.berkeley.edu/$\sim$nuzzo/Research.html.
[26] “3-sat,” http://people.cs.ubc.ca/$\sim$hoos/SATLIB/benchm.html.
[27] C. Tomlin, G. Pappas, and S. Sastry, “Conflict resolution for air traffic management: A study in multi-agent hybrid systems,” *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 110–120, Apr. 1998.
[28] T. Vo, E. Barr, and Z. Su, personal communication.