Abstraction Based Theorem Proving: An example from the theory of Reals

Ashish Tiwari*

SRI International,
333 Ravenswood Ave,
Menlo Park, CA, U.S.A
Tel:+1.650.859.4774
Fax:+1.650.859.2844
tiwari@csl.sri.com

Abstract. This paper presents a sound, but incomplete, logical decision procedure for the quantifier-free theory of reals. It is obtained using a combination of superposition and ordered chaining calculi. The emphasis is on termination and efficiency, rather than completeness. The procedure is used at the core of an hybrid abstraction tool for creating finite state sound abstractions of hybrid systems.

1 Introduction

Soundness, completeness, and termination are three crucial properties of inference systems. If an inference system is sound and complete with respect to a logical theory and terminating as well, then it can be used to obtain a decision procedure for that theory. However, many logical theories, including the full first-order theory, are undecidable. For an undecidable theory T, any inference system will either not be sound and complete with respect to T, or it will be non-terminating.

In light of the undecidability, the emphasis in the theorem proving community has been on designing sound and complete, but nonterminating, inference systems. In this paper, we follow a different paradigm. Our interest is in sound and terminating inference systems, that may not really be complete for the given theory of interest.

One can argue that an inference system that is not sound and complete with respect to a theory T of interest, is not really an inference system *for* T. Indeed, given an inference system I which is sound with respect to T and terminating, we can define a theory T' with respect to which it is also complete. Here T' is defined to be the set of (the negation of) all sentences which are (shown contradictory) proved by I. Of course, for "weak" inference systems I, T' might not even be

* Research supported in part by DARPA under the MoBIES and SEC programs administered by AFRL under contracts F33615-00-C-1700 and F33615-00-C-3043, respectively, and NASA Langley Research Center contract NAS1-00108 to Rannoch Corporation.

deductively closed. But, due to soundness, $T' \subseteq T$ and we call any such T' an *abstraction* of the theory T.

Given a fixed theory T of interest, there are several abstractions T' of T for which sound, complete, and terminating inference systems can be designed. Any set of sound (with respect to T) and terminating inference system defines, as we saw above, an abstraction T' of T. This is also true of abstractions of state transition systems, as is popular in formal verification. The important question, therefore, in both areas, is really the identification of *pragmatically useful* abstractions.

In this paper, the theory T of interest is the quantifier-free theory of reals. We present an inference system which is sound with respect to T and also terminating. However, the abstract theory T' with respect to which it is also complete, is hard to characterize in any other way.

1.1 Motivation

We briefly motivate why identifying such pragmatically useful decidable abstractions T' of logical theories T are of interest. Many applications use reasoning systems in a *failure-tolerant* mode, that is, failure to identify an unsatisfiable formula (as being unsatisfiable) is not catastrophic to the application. For example, in compiler optimization, the ability to perform nontrivial reasoning only results in better optimized code. Failure of the reasoning system does not interfere with the correctness of the compiler itself. As a second example, consider the process of constructing (predicate) abstractions of state transition systems. An incomplete prover, again, only results in a coarser abstractions. Any abstraction constructed by such a prover, would still be a "sound" abstraction. In both these applications, note further that the theorem prover is embedded inside the application, and hence termination is clearly more desirable.

A second motivation comes from the observation that very often the full theory of interest is either undecidable, or has a decision procedure with a very large computational complexity. In either case, it is important to identify pragmatically useful abstractions T', or in other words, identify sound, but incomplete, and terminating inference systems.

2 The theory of Reals

We assume the reader is familiar with the definition of a field K over a signature consisting of binary symbols $+$ and $\cdot$, unary symbols $-$ and $^{-}$, and constants 0 and 1. A real closed field is formally defined as follows.

Definition 1. *An ordered field K is a field together with a subset P of K of positive elements such that (i) $0 \notin P$, (ii) if $a \in K$, then either $a \in P$, $a = 0$, or $-a \in P$, and (iii) P is closed under the addition $+$ and multiplication $\cdot$ operators.*

A real closed field K is an ordered field such that (i) every positive element of K has a square root in K and (ii) every polynomial $f(x)$ in $K[x]$ of odd degree has a root in K.

The theory of real closed fields, RCF, is defined over the signature $\langle \mathbb{Q}, +, -, \cdot, >, \geq, \approx \rangle$, where $\mathbb{Q}$ is the set of rational numbers, $+, \cdot$ are binary function symbols, $-$ is a unary symbol, and $>, \approx, \geq$ are binary predicates. The field of real numbers $\mathbb{R}$ is a model of RCF, but it is not the only model. However, it can be shown that the full first-order theories $Th(\mathbb{R})$ and $Th(RCF)$ are identical.

The first-order theory of real closed fields was shown to be decidable by Tarski [13] by showing that the theory admitted quantifier elimination. This theory has a known upper-bound of $n^{2^{O(n)}}$, thanks to Collin's method based on cylindrical algebraic decomposition [5]. Over the last 25 years, the CAD algorithm has been improved and made more efficient [9, 11, 12].

Although an excellent implementation of the cylindrical algebraic decomposition based decision procedure is available [10], the double exponential complexity for the decision procedure greatly hampers its utility for applications alluded to above. Moreover, in our application to automatically construct abstractions for hybrid systems, the proof obligations are mostly from the *quantifier-free* theory of the reals and they are often just a *conjunction of atomic formulas.*

We describe below the procedure that has worked best in our application domain. It is a *logical* decision procedure obtained by combining the superposition calculus for constructing Gröbner bases of polynomial equations [3], with an ordered chaining calculus for handling inequalities. Ordered chaining calculi for abstract transitive relations have been developed previously [2]. This paper uses them in the specific context of the greater-than $>$ and greater-than-equal-to $\geq$ relation over the reals.

3 Polynomials

Terms over the signature of reals can be represented as *polynomials*, written as $\Sigma_{1 \leq i \leq n} c_i t_i$, where c_i's are elements of $\mathbb{Q}$ and t_i's are products over variables, called *power-products.* The computation of a polynomial form is best described via a convergent rewrite system over the following sorted signature.

$$\begin{array}{rrl}
0, 1, \ldots, c \in \mathbb{Q}: & & \to \mathit{Coef} \\
-: & \mathit{Coef} & \to \mathit{Coef} \\
+, \times: & \mathit{Coef} \times \mathit{Coef} & \to \mathit{Coef} \\
I, x_1, \ldots, x_n: & & \to \mathit{PProd} \\
\cdot: & \mathit{PProd} \times \mathit{PProd} & \to \mathit{PProd} \\
M: & \mathit{Coef} \times \mathit{PProd} & \to \mathit{Poly} \\
\mathbf{0}: & & \to \mathit{Poly} \\
\oplus, \otimes: & \mathit{Poly} \times \mathit{Poly} & \to \mathit{Poly}
\end{array}$$

Terms of sort *Coef*, *PProd*, and *Poly* are respectively called coefficients, power-products, and polynomial expressions. We use u, v for variables of sort *Coef*, μ, ν for *PProd* variables, and α, β, γ for *Poly* variables. Every term over the signature of the theory of reals can be represented as a term of sort *Poly*. Terms of sort *Poly* can be normalized via the following rewrite system, which is ground convergent modulo the associativity and commutativity axioms (AC), to give the corresponding polynomial form for the term [3].

$$\mu \cdot I \to \mu \tag{1}$$
$$M(0, \nu) \to \mathbf{0} \tag{2}$$
$$\mathbf{0} \oplus \alpha \to \alpha \tag{3}$$
$$\mathbf{0} \otimes \alpha \to \mathbf{0} \tag{4}$$
$$M(u, \nu) \oplus M(v, \nu) \to M(u + v, \nu) \tag{5}$$
$$M(u, \mu) \otimes M(v, \nu) \to M(u \times v, \mu \cdot \nu) \tag{6}$$
$$\alpha \otimes (\beta \oplus \gamma) \to (\alpha \otimes \beta) \oplus (\alpha \otimes \gamma) \tag{7}$$

The equational theory induced by the above rewrite system (modulo AC) is identical to the equational theory of polynomials [3]. Note that the variables $x_1, \ldots, x_n$ are treated as special constants for purposes of computing normal forms for *Poly* terms using the above rewrite system. "Ground" terms (of sort *Poly*) in normal form correspond to (and will be called) *polynomials*. They can be written as sums of monomials $M(c_1, t_1) \oplus M(c_2, t_2) \oplus \cdots M(c_n, t_n)$, which will be notationally simplified to $\Sigma_j c_j t_j$. The above sorted signature was introduced purely for the purpose of presenting the convergent rewrite system for normalizing terms in the signature of reals. In the subsequent sections, we will use the popular and simplified notation for writing polynomials. In particular, the symbol $\mathbf{0}$ will be represented by 0, $\oplus$ will be written as $+$, and M and $\otimes$ will be replaced by juxtaposition. We will use (meta-variables) c, d to denote elements of $\mathbb{Q}$, s, t to denote ground *PProd* terms, and p, q, r to denote ground *Poly* terms.

4 A decision procedure for an abstract theory of reals

Our algorithm maintains ground *PProd* terms p in polynomial normal-form and atomic formulas in the form $p \approx 0$, $p > 0$, or $p \geq 0$. A conjunction of such atomic formulas is described as a tuple (E, R, S), where E, R and S contain atomic formulas of the first, second, and third kinds respectively. Since interest is in the satisfiability of the conjunction of these atomic formulas, the variables can be treated as *constants*, as in [3], and an ordering $\succ$ defined on the power-products, that can be suitably extended to polynomials. More specifically, if $\succ$ is any precedence on the indeterminates $x_1, \ldots, x_n$, then we have a choice between using "lexicographic ordering" or "total degree lexicographic ordering" on power-products [3,4]. For example, if we choose the precedence $x_1 \succ x_2 \succ x_3$ over the 3 variables, then $x_1 x_2^2 \succ x_1^2$ in the total-degree lexicographic ordering (since degree of $x_1 x_2^2$ is 3, which is greater than the degree of x_1^2), while x_1^2 is greater in the lexicographic ordering since degree of x_1 is greater in x_1^2.

Our inference rules work on the highest power-product in a polynomial, and hence we often write a polynomial p as $c_0 t_0 + \Sigma_{1 \leq i \leq n} c_i t_i$ to distinguish the highest power-products t_0 from the rest. For example, in the polynomial $2x_1^2 x_2 + x_2^2$, $c_0 = 2$ and $t_0 = x_1^2 x_2$, if we use the total degree lexicographic ordering.

4.1 Gröbner Basis for Polynomial Equations

Polynomial equations are handled by using the completion-like method for constructing Gröbner bases [3]. Since the coefficient domain is restricted to rationals, the computation is much simplified. For purposes of efficiency, though, the full blown Gröbner basis computation is (optionally) replaced by a computation that only performs *simplifications* and avoids all *deduction* steps.

Let $q = d_0t_0 + \Sigma_i d_i t_i$ be a polynomial with leading power-product t_0. A polynomial equation $q = 0$ can be used as a rewrite rule to simplify other polynomials. If $p = p' + c_k s_k$ such that s_k can be written in the form $t_0 t_0'$, then we define $p \rightarrow_{q=0} p' + p''$, where $p'' = c_k s_k + (-c_k t_0'/d_0)(d_0 t_0 + \Sigma_i d_i t_i)$. Note that p will be simplified into a polynomial form using the convergent rewrite system from Section 3 before it is added in the consequent below.

$$\frac{(E \cup \{p \approx 0,\ q \approx 0\}, R, S)}{(E \cup \{p' \approx 0,\ q \approx 0\}, R, S)} \qquad \texttt{if } p \rightarrow_{q\,\approx\,0} p'$$

$$\frac{(E\{q \approx 0\}, R \cup \{p > 0\}, S)}{(E \cup \{q \approx 0\}, R \cup \{p' > 0\}, S)} \qquad \texttt{if } p \rightarrow_{q\,\approx\,0} p'$$

$$\frac{(E\{q \approx 0\}, R, S \cup \{p \geq 0\})}{(E \cup \{q \approx 0\}, R, S \cup \{p' \geq 0\})} \qquad \texttt{if } p \rightarrow_{q\,\approx\,0} p'$$

It is clear that repeated applications of the above inference rules is bound to terminate: the cardinality of the E, R, S sets does not increase and terms in the sets E, R, S only get smaller in the well-founded ordering $\succ$. When restricted to the equational subcase, these inference rules capture the *collapse* rule, but do not compute nontrivial critical pairs (that is, do not capture the *superposition* rule below), and hence, do not result in a convergent set (Gröbner basis) for the polynomial equations.

$$\frac{(E' = E \cup \{c_0 s_0 + p \approx 0,\ d_0 t_0 + q \approx 0\}, R, S)}{(E' \cup \{c_0 t_0' q - d_0 s_0' p \approx 0\}, R, S)} \qquad \texttt{if } t_0' t_0 = s_0' s_0 = lcm(s_0, t_0) \neq s_0 t_0$$

The full Gröbner basis computation for the equations E is available as an option, but in its default setting, this is turned off. Note that a full Gröbner basis computation is "complete" only for algebraic closed fields, and hence doing the superposition inferences is still not complete for real closed fields. For completeness, among other things, we would need additional rules for polynomial factorization.

Example 1. Let $E = \{x_1^2 x_2 \approx 0,\ x_1 x_2 \approx 1\}$, $R = S = \emptyset$. Now, $x_1^2 x_2$ rewrites to $1x_1$, which normalizes to x_1, by the equation $x_1 x_2 \approx 1$. Hence, using the first inference rule, we can deduce the new equation $x_1 \approx 0$. Now, $x_1 x_2$ can be simplified to $0x_2$, which normalizes to 0, by $x_1 \approx 0$. Thus, we infer $1 = 0$ and add it to the set E. The inconsistency detection rules in Section 4.3 detect this as an inconsistency.

4.2 Ordered Chaining for Polynomial Inequalities

We use an ordered chaining calculus, inspired by the work of Bachmair and Ganzinger [2], to deal with the inequalities.

Definition 2. *Let $p = c_0 s_0 + p'$ be a polynomial with leading power-product s_0 and leading coefficient c_0. Let P be a set of polynomials. Let $q_1, q_2, \dots, q_k \in P$ be such that the product of leading power-products of q_i's is equal to s_0, and the product of leading coefficients of q_i's is d_0. If $c_0 d_0 < 0$, then we say that the polynomial $p + |\frac{c_0}{d_0}| q_1 q_2 \cdots q_k$ is a* k-chaining of p and P.

If the polynomial p and each polynomial in P is known to be non-negative, then the k-chaining polynomial of p and P will also be non-negative. If p is strictly positive, then the k-chaining polynomial will also be strictly positive. This reasoning is used for dealing with inequalities and formally it is stated as the following inference rules.

$$\frac{(E, R \cup \{p > 0\}, S)}{(E, R \cup \{p > 0, q > 0\}, S)} \quad \texttt{if } q \text{ is a } k\text{-chaining of } p \text{ and } Pols(R \cup S \cup E \cup E^-)$$

$$\frac{(E, R, S \cup \{p \geq 0\})}{(E, R, S \cup \{p \geq 0, q \geq 0\})} \quad \texttt{if } q \text{ is a } k\text{-chaining of } p \text{ and } Pols(R \cup S \cup E \cup E^-)$$

$$\frac{(E, R, S \cup \{p \geq 0\})}{(E, R \cup \{q > 0\}, S \cup \{p \geq 0\})} \quad \texttt{if } q \text{ is a } k\text{-chaining of } p \text{ and } Pols(R)$$

Here $Pols(ERS) = \{p : p \rhd 0 \in ERS, \rhd \in \{\approx, >, \geq\}\}$ and $E^- = \{-p \approx 0 : p \approx 0 \in E\}$. In affect, each equation $p \approx 0$ is being treated here as the conjunction of two inequalities $p \geq 0$ and $-p \geq 0$.

Note that k is a parameter in the inference rules above. The instances where $k = 1$ are applied preferably. Again, for purposes of efficiency, the implementation only considers k-chaining for $k \leq 3$.

Example 2. Let $R = \{-x_1 x_2 + 5 > 0,\ x_1 - 2 > 0,\ x_2 - 4 > 0\}$ and $E = S = \emptyset$. The polynomial $-x_1 x_2 + 5 + (x_1 - 2)(x_2 - 4)$, which normalizes (by the convergent rewrite system of Section 3) to $-4x_1 - 2x_2 + 3$, is a 2-chaining of $-x_1 x_2 + 5$ and $Pols(R)$. Thus, we deduce the inequality $-4x_1 - 2x_2 + 3 > 0$. A 1-chaining of this new inequality and $Pols(R)$ gives $-4x_1 - 2x_2 + 3 + 4(x_1 - 2) > 0$, which simplifies to $-2x_2 - 5 > 0$. A yet another 1-chaining results in $-2x_2 - 5 + 2(x_2 - 4) > 0$, which normalizes to $-13 > 0$, which is detected as inconsistent by Section 4.3 inference rules.

An important point to note here is that $k = 1$ is sufficient for *linear* formulas, which is an efficiently decidable fragment of the quantifier-free theory of the reals. The specialization of the first of these three inference rules for the linear case can be written as:

$$\frac{(E, R' = R \cup \{c_0 x + p > 0,\ -d_0 x + q > 0\}, S)}{(E, R' \cup \{d_0 p + c_0 q > 0\}, S)} \quad \texttt{if } c_0 > 0, d_0 > 0, \{x\} \succ^m \{p, q\}$$

Note that the ordering restriction constrains x to be the highest precedence variable. In this linear case, the decision procedure obtained using the above inference rules mimics the well-known Fourier-Motzkin decision procedure for linear rational arithmetic [6]. For nonlinear formulas too $k = 1$ is tried in preference to higher values for k. In this way, even on nonlinear formulas, our decision procedure first checks whether it can detect possible inconsistencies by "ignoring the nonlinearity", or in other words, abstracting the nonlinear parts by new "linear" terms (implicitly).

The chaining inference rules increase the cardinality of $R \cup S$. Note, however, that the deduced polynomial is smaller than the original polynomial $c_0 s_0 + p$ (in the ordering $\succ$). But still, the set of inference rules described above do *not* terminate. This was a surprising result, but it is a consequence of the fact that the coefficient domain, the set of rational numbers, is not finite. The procedure emulates certain iterative root finding algorithms from numerical analysis.

Example 3. Let $R = \{-x^2 + 4x - 4 > 0, x > 0\}$, $E = \emptyset$, and $S = \emptyset$. A 2-chaining of $-x^2 + 4x - 4$ and $Pols(R)$ is the polynomial $4x - 4$, and hence we deduce $4x - 4 > 0$, or $x - 1 > 0$. Now, a 2-chaining polynomial of $-x^2 + 4x - 4$ and $Pols(R \cup \{x - 1 > 0\})$ is the polynomial $2x - 3$. Thus, starting with $x > 0$, we have now obtained $x > 1$ and $x > 3/2$. The next iteration of inference gives $-x^2 + 4x - 4 + (x - 3/2)(x - 3/2) > 0$ which simplifies to $x > 7/4$.

Continuing this way, from $-x^2 + 4x - 4 > 0$ and $x > c$, we can deduce $x > c'$ where $c' = (4 - c^2)/(4 - 2c)$. The value of c' is always less than 2. Hence, we can deduce formulas of the form $x > c$, with c getting successively closer to 2, but never really becoming 2. Thus, the inference rules do not terminate and fail to detect the inconsistency of the original inequalities.

The inference system is clearly sound. We force termination by explicitly bounding the number inferences in a single run of the saturation procedure. We also use strong heuristics and deletion of redundant rules in the procedure. In most examples, the strong heuristics and redundancy elimination rules also ensure termination of the saturation procedure. But, in some rare cases, forcing an explicit termination is required.

4.3 Inconsistency and Redundancy

The decision procedure terminates when it detects an inconsistency. The following inference rules detect trivial inconsistencies.

$$\frac{(E \cup \{c_0 \approx 0\}, R, S)}{\perp} \qquad \texttt{if } c_0 \neq 0$$

$$\frac{(E, R \cup \{c_0 > 0\}, S)}{\perp} \qquad \texttt{if } c_0 \leq 0$$

$$\frac{(E, R, S \cup \{c_0 \geq 0\})}{\perp} \qquad \texttt{if } c_0 < 0$$

Removal of redundant rules is crucial for efficiency of the decision procedure. Equations of the form $0 \approx 0$, and strict inequalities of the form $c > 0$, where c is indeed greater than 0, and inequalities of the form $c \geq 0$, where c is nonnegative are redundant and can be deleted. However, we can a use stronger definition of redundancy and delete more rules.

The removal of redundant equations is implicit in the formulation of the inference rules above. The simplification rules introduce new equations and delete the old redundant equation simultaneously. Hence, we only need to define the notion of redundancy on inequalities.

Definition 3. *Trivial inequalities of the form $c > 0$ (respectively $d \geq 0$) are redundant if the constant c (d) is positive (nonnegative). A rule $q > 0$ (respectively $r \geq 0$) is redundant in the state $(E = \{p_i \approx 0 : i = 1, 2, \ldots\}, R = \{q_j > 0 : j = 1, 2, \ldots\}, S = \{r_k \geq 0 : k = 1, 2, \ldots\})$ if $q \rightarrow_E^* q'$ and $q' = \Sigma_j c_j q_j + \Sigma_k d_k r_k + e$ for some nonnegative constants c_j's and d_k's, and positive constant e.*

The following two inference rules remove redundant inequalities.

$$\frac{(E, R \cup \{p > 0\}, S)}{(E, R, S)} \qquad \texttt{if } p > 0 \text{ is redundant in } (E, R, S)$$

$$\frac{(E, R, S \cup \{p \geq 0\})}{(E, R, S)} \qquad \texttt{if } p \geq 0 \text{ is redundant in } (E, R, S)$$

Note that strong notions of redundancy can possibly cause incompleteness in deductive systems. But since we make no claims about completeness, the "correctness" of the definition of redundancy is really vacuous. Nevertheless, we briefly motivate Definition 3 here. As described above, an equation $c_0 s_0 + p \approx 0$ in E can be used as rewrite rule $c_0 s_0 \rightarrow -p$. The inequalities in $R \cup S$ can also be used as rewrite rules in the same way, but restricted somehow. Unlike the semantic equality relation $\approx$ over reals, the semantic relations $>$ and $\geq$ are not *congruences* and hence these rewrite rules cannot be used inside arbitrary contexts. In particular, if $c_0 s_0 > \Sigma_i c_i s_i$, (equivalent representation for $c_0 s_0 + \Sigma_i - c_i s_i > 0$) then it is not the case that $c_0 s_0 q > \Sigma_i c_i s_i q$, for arbitrary polynomial q. But this is true for some restricted q, for example, when q is a positive rational. Thus, if we use $R \cup S$ inequalities as rewrite rules, but restricted to within the context of a positive rational, and use the equations in E in the usual way, then we can construct *rewrite derivations*, or a *proof* of inequalities $q > 0$ (or $r \geq 0$). Now, an inequality $q > 0$, or $r \geq 0$, is defined as redundant in Definition 3 if there is such a rewrite proof for it using $E \cup R \cup S$ and the trivially redundant inequalities. In the spirit of proof simplification and proof orderings [1], we can view the process of saturation as simplifying proofs until they are all in some normal form–valley proofs for equations and rewrite proofs for inequalities. Thus, any inequality which already has a simpler normal form proof using existing other inequalities and equations can be deleted.

Definition 3 is not easy to apply in its general form. Therefore, just like the k-chaining inference rule, we only search for restricted kinds of redundant inequalities. More specifically, we first fully simplify q by the equations in E and then check if the resulting polynomial, say q', can be written in one of the following forms: e, $c_j q_j + e$, $d_k r_k + e$, or $c_j q_j + d_k r_k + e$ for some j, k, and positive (or, nonnegative, depending on the case) e.

Example 4. The inequality $2x_1x_2 - 2x_1 - 5 > 0$ is redundant in the state $(\emptyset, R = \{x_1x_2 - 5 > 0,\ -x_1 + 2 > 0\}, \emptyset)$ because $2x_1x_2 - 2x_1 - 5 = 2(x_1x_2 - 5) + 2(-x_1 + 2) + 1$ and $1 > 0$.

4.4 Heuristics

We use some additional heuristics to limit the search space (the number of new rules in the sets R and S). The most crucial heuristic currently in use limits the number of inequalities with the same leading power-product and identical coefficient sign that can be present simultaneously in the sets R and S. In the default setting, this limit is set to 2.

$$\frac{(E, R \cup \{c_0 s_0 + p > 0,\ c_0' s_0 + p' > 0,\ c_0'' s_0 + p > 0\}, S)}{(E, R \cup \{c_0 s_0 + p > 0,\ c_0' s_0 + p' > 0\}, S)} \quad \texttt{if } \mathit{sign}(c_0) = \mathit{sign}(c_0') = \mathit{sign}(c_0'')$$

$$\frac{(E, R, S \cup \{c_0 s_0 + p \geq 0,\ c_0' s_0 + p' \geq 0,\ c_0'' s_0 + p \geq 0\})}{(E, R, S \cup \{c_0 s_0 + p \geq 0,\ c_0' s_0 + p' \geq 0\})} \quad \texttt{if } \mathit{sign}(c_0) = \mathit{sign}(c_0') = \mathit{sign}(c_0'')$$

If one of the rules is redundant, then it is deleted. But if none is redundant, then the choice of rule to be removed (in the above inference rule) is arbitrary.

The decision procedure also uses a special inference rule based on symbolic computation of upper and lower bounds of polynomial expressions. If numerical lower and upper bounds of certain variables are known, then symbolic lower and upper bounds of polynomial expressions can be computed, which would have a lower degree than the original polynomial.

$$\frac{(E, R \cup \{p + c x_i \mu > 0\}, S)}{(E, R \cup \{p + c x_i \mu > 0, p + c d_i \mu > 0\}, S)} \quad \texttt{if } c > 0, (R \cup S) \models 0 \leq x_i \leq d_i, (R \cup S) \models \mu > 0$$

The checks $(R \cup S) \models 0 \leq x_i \leq d_i$ and $(R \cup S) \models \mu > 0$ are done by simply going through the set $R \cup S$ and maintaining a numerical lower and upper bound for x_i and each variable in the power-product μ. The rule can be suitably modified if the coefficient c is negative, or when the inequality is in the set S.

The inference system is clearly sound. Every rule or equation in the sets E, R, and S is a consequence (in the theory of reals) of the formulas in the initial set.

Proposition 1. *Let (E, R, S) be an initial state consisting of equations E and inequalities R and S. If $\perp$ can be reached from the initial state using the given inference rules, then the set of atomic formulas $E \cup R \cup S$ (interpreted as a conjunction) is unsatisfiable in the theory of reals.*

As mentioned above, termination is explicitly enforced. The inference systems is clearly not complete, as shown by Example 3.

5 Features of the Decision Procedure

Some of the unique and important features of the decision procedure described here are that it is logical, incremental, and produces witnesses. Having a logical decision procedure is important for purposes of integration with other decision procedures and other theorem proving components.

Saturation based procedures are incremental. Equations and inequalities are asserted one at a time in a context specified by the triple (E, R, S). The result of the assertion is a new triple (E', R', S') (or $\perp$ and a witness, see below) that is obtained by saturating the existing set (E, R, S) and the new assertion under the above inference rules. Only inferences with the newly asserted fact need to be computed.

5.1 Witness

The decision procedure also keeps track of witnesses or proofs for derived formulas. Each atomic formula is represented as a structure containing a polynomial, an operator ($\approx$, $>$, or $\geq$), and a *witness*. A witness is a list of pointers to other atomic formulae in the set $E \cup R \cup S$. Each application of a (deductive) inference rule creates a new atomic formula and stores a witness with it (a pointer to the atomic formulas in the antecedent of the rule).

The witness facility is primarily used by the feasibility checker of the hybrid abstractor, which is described in detail below.

6 Application Domain

The set of inference rules described in Section 4 are used to obtain a terminating procedure for deciding satisfiability of conjunctions of atomic formulas over the theory of reals. Presently, the decision procedure is used as a black box, embedded inside an automated abstraction tool for hybrid systems. The proof obligations generated by the hybrid abstraction tool range from fairly simple obligations to quite complex formulas depending on the particular hybrid system specification. They could be nonlinear in general.

The hybrid abstractor uses the decision procedure in two different ways. First, proof obligations are generated in the process of abstracting the (continuous and discrete) dynamics of the (hybrid) system. When restricted to discrete transition systems, the proof obligations are similar to those generated by a predicate abstraction tool [8]. The proof obligations arising from abstracting the continuous dynamics are similar too.

The hybrid abstractor also uses the decision procedure to identify which abstract states are actually feasible. This step, which can be obliterated while constructing discrete transition system abstractions [8], becomes crucial for hybrid systems. The witness generator is used here for efficiency. When an abstract state is identified as infeasible (by the decision procedure), the decision procedure also returns a witness for the inconsistency and thus the abstraction tool can infer infeasibility of a *set* of abstract states. The check for infeasibility can be done at the time of constructing the abstraction, or during model-checking [15]. For examples of hybrid systems that are handled by the hybrid abstractor and decision procedure, the reader is referred to [7, 14].

Compared to the QEPCAD tool, which we first used for experiments, the new sound, but incomplete, procedure is faster and more efficient. We present here some results of experiments comparing the QEPCAD tool with our decision procedure. All test examples consisted of conjunction of atomic formulas, with all variables assumed to be existentially quantified. Table 1 gives the example formulas and Table 2 gives the running time of the QEPCAD procedure and the decision procedure (DP) described in this paper. DP correctly decides each of the formulas in this test bench and is considerably faster. QEPCAD (Version 13) fails on two. All experiments were done on a personal desktop i686 machine running Linux.

	Atomic formulas
Test1	$\{x_c - x_d > 0, x_c > 0, x_d > 0, x_a > 0, x_b > 0, x_a > 0, x_a x_b + x_a x_c - 10 < 0\}$
Test2	$\{2a_p - 3 = 0, 2b_p - 1 > 0, 4a - 1 > 0, 1 - 2e > 0, v7 > 0, v_{ba} > 0, v_{bap} > 0, 3v_{ba} - 8v_7 - 8v_{bap} > 0, ap \geq 0, bp \geq 0, a \geq 0, e \geq 0\}$
Test3	$\{r_a + 5r_b^2 - 20r_b = 0, 25r_b^3 - 100r_b^2 + 20r_b - 1 = 0, r_a > 0, r_b > 0, 4 - v > 0, v + 4 > 0, 1 - a > 0, h - 1 > 0, a + 1 > 0, 10 - g > 0, g - 4a - 4v > 0, 1 - 2h > 0, g - r_a v - r_b a = 0\}$
Test4	$\{l^3 + 3l^2 + 4l + 1 = 0, r_a = l^2 + 3l^2, r_b = l, 0 < 2l + 1, 4l + 1 < 0, v_f > 0, v > 0, 0 < a + 2, v_f - v > 0, g + 10 > 0, g - r_a v_f + r_a v - v_f + r_b a > 0\}$
Test5	$Test4 \cup \{g + 10 < 0\}$

Table 1. Atomic formulas for the test examples.

The experimental results reported in Table 2 comparing the two decision procedures should be interpreted in the right context. First, the test bench consists of formulas of a very restricted kind. The QEPCAD decision procedure is much more general and can handle arbitrary first-order formulas, whereas DP cannot. Furthermore, it is quite easy to create very small examples to expose the incompleteness of DP, while QEPCAD works just fine on them. Second, the QEPCAD decision procedure was run in its interactive mode, that is, the input was typed in by hand, in the input format of QEPCAD. The DP procedure is written in Lisp and the input was provided in Lisp. The decision procedure code was compiled and loaded into Lisp.

But despite the unfairness, it is amply clear that QEPCAD is not well suited for applications where termination and soundness are essential, and "logical" aspects are crucial. Test 5 combines a trivial inconsistency $gap + 10 > 0$, $gap + 10 < 0$ with a large formula. But, QEPCAD fails on this "logically simple" example.

	#variables	degree	status	QEPCAD	DP
Test1	4	2	satisfiable	30	1
Test2	7	3	inconsistent	20	1
Test3	6	3	inconsistent	4240	2
Test4	7	3	inconsistent	FAIL	5
Test5	7	3	inconsistent	FAIL	5

Table 2. Total running time averages (in milliseconds) for Tests1–5.

7 Conclusion

Motivated by applications that need embedded symbolic reasoning component (for *enhancing* their quality), we argue for the need for abstraction based theorem proving. The correct abstraction (degree of incompleteness) is application dependent and it is a question of pragmatics as it offers a time versus accuracy tradeoff. In this article, we investigated a sound, incomplete, and terminating procedure for checking satisfiability of formulas over the theory of real closed fields. This has proved very useful for the application domain of hybrid abstraction.

As part of future work, it will be interesting to study additional inference rules that can be added to the above set without compromising efficiency. Characterizing the theory for which such a procedure is complete is also intriguing. One also wonders if the current set of inference rules can be extended to give a *complete* and logical decision procedure for the quantifier-free theory of reals.

The long term goal of this research is to obtain an efficient and practical implementations of many different *logical* algorithms for deciding (certain classes of) formulas over the theory of real closed fields. We envision a collection of procedures that successively decide larger and larger fragments of the theory and that interact with each other based on abstraction and refinement.

Acknowledgements. The author wishes to thank Hassen Saïdi for help in concretizing some ideas presented in this paper.

References

1. L. Bachmair. *Canonical Equational Proofs*. Birkhäuser, Boston, 1991.
2. L. Bachmair and H. Ganzinger. Rewrite techniques for transitive relations. In *9th Annual IEEE Symposium on Logic in Computer Science*, pages 384–393, 1994.
3. L. Bachmair and A. Tiwari. D-bases for polynomial ideals over commutative noetherian rings. In H. Comon, editor, *Rewriting Techniques and Applications, RTA 1997*, volume 1103 of *Lecture Notes in Computer Science*, pages 113–127, Sitges, Spain, July 1997. Springer-Verlag.
4. T. Becker and V. Weispfenning. *Gröbner bases: a computational approach to commutative algebra*. Springer-Verlag, Berlin, 1993.

5. G. E. Collins. Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In *Proc. Second GI Conf. Automata theory and formal languages*, pages 134–183, 1975. Vol. 33 of *Lecture Notes in Comp. Sci., Springer, Berlin*.
6. G. B. Dantzig and B. C. Eaves. Fourier-motzkin elimination and its dual. *J. of combinatorial theory (A)*, 14:288–297, 1973.
7. R. Ghosh, A. Tiwari, and C. Tomlin. Automated symbolic reachability analysis with application to delta-notch signaling automata. In O. Maler and A. Pnuelu, editors, *Hybrid Systems: Computation and Control HSCC*, volume 2623 of *LNCS*, pages 233–248. Springer, April 2003.
8. S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In O. Grumberg, editor, *Proc. of 9th Conference on Computer-Aided Verification (CAV'97)*, volume 1254, pages 72–83. Springer Verlag, 1997.
9. H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proc. ISAAC 90*, pages 261–264, 1990.
10. H. Hong. Quantifier elimination in elementary algebra and geometry by partial cylindrical algebraic decomposition version 13. In *The world wide web*, 1995. `http://www.gwdg.de/~cais/systeme/saclib, www.eecis.udel.edu/~saclib/`.
11. D. Lazard. *An improved projection for cylindrical algebraic decomposition*. Technical Report, Informatique, Universite Paris IV, F-75252 Paris Cedex 05, France, 1990.
12. S. McCallum. An improved projection operator for cylindrical algebraic decomposition of three dimensional space. *J. Symbolic Computation*, 5:141–161, 1988.
13. A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1948. Second edition.
14. A. Tiwari. Approximate reachability for linear systems. In O. Maler and A. Pnuelu, editors, *Hybrid Systems: Computation and Control HSCC*, volume 2623 of *LNCS*, pages 514–525. Springer, April 2003.
15. A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In C. J. Tomlin and M. R. Greenstreet, editors, *Hybrid Systems: Computation and Control, HSCC 2002*, volume 2289 of *LNCS*, pages 465–478, 2002.