Solving Constraints for an Instance of an Extended CLP Language over a Domain Based on $\mathcal{R}$eal Numbers and $\mathcal{H}$erbrand Terms

Miguel García-Díaz and Susana Nieva *
Dpto. Sistemas Informáticos y Programación
Universidad Complutense de Madrid
e-mail:{miguel, nieva}@sip.ucm.es

Abstract

Combining the logic of hereditary Harrop formulas *HH* with a constraint system, a logic programming language is obtained that extends Horn clauses in two different directions, thus enhancing substantially the expressivity of Prolog. This new language is parametric over the constraint system. A constraint solver for a particular instance will test the satisfiability of formulas built up by means of terms and predicates belonging to the particular domain, and by the connectives and quantifiers usual in first-order logic. We introduced a useful constraint system called $\mathcal{RH}$, which is a hybrid domain mixing Herbrand terms and real numbers. For such system, a procedure to simplify constraints was provided. It is based on the combination of two different eliminating quantifiers mechanisms, one used for solving unification and disunification problems, the other used to solve the part of the formulas that only contains real predicates. In this paper we review and extend the presentation of the language as well as of the simplification procedure. Moreover a novel algorithm to check the satisfiability of a class of simplified constraints is introduced. In this way, we have provide a procedure to solve $\mathcal{RH}$-constraints in the context of *HH* with constraints.

Keywords: constraint systems, hereditary Harrop formulas, real numbers, finite symbolic trees, first-order logic.

1 Introduction

Constraint logic programming *CLP* [10] arises as an extension of the traditional logic programming, in which only Horn clauses are allowed, with the purpose of overcoming its inherent limitations in dealing efficiently with elements of domains different from Herbrand terms. In order to build an implementation, it is necessary to find a combination of the goal solving procedure, concerning the logic part of the language (e.g. *SLD*-resolution [14] for Horn clauses), with a decision mechanism for constraint satisfiability, capable of generating a set of answers that somehow captures the whole set of valid answer constraints for any goal, which will depend on the specific domain of the constraint system.

*The authors have been supported by the Spanish National Project TIC 2002-01167 *MELODIAS*

This kind of extension is also dealt with in [13, 12]. However, in both cases, the base language is the logic of Hereditary Harrop formulas *HH* (first-order and higher-order, respectively). Such logic is in addition an extension of the logic of Horn clauses in which implications and universal quantification are allowed in goals [16]. *SLD*-resolution, which deals with equality between Herbrand terms by means of unification, is no longer a choice for this extension, mainly due to the existence of prefixes with mixed quantifications, i. e. prefixes containing both $\exists$ and $\forall$ arbitrarily ordered. In [17], a method based on labeled unification is used for *HH*. However, in *HH(C)*, which is the extension of *HH* including constraints, the technique to obtain answer substitutions by means of unification is replaced by a test of the satisfiability of constraints under a mixed prefix. As a consequence, the specific mechanism to decide the satisfiability should be able to deal with complex formulas, which may contain occurrences of any connective in first-order logic, equations between Herbrand terms and, moreover, other relations belonging to the constraint signature applied to terms of the instance domain.

Following one of the open research lines mentioned in [13], we have searched for decidable theories which lead to useful first-order constraint systems, in order to use them as instance domains for the abstract logic programming language *HH(C)*. The natural domain, traditionally used in declarative programming, is the Herbrand universe, which consists of symbolic terms built up with constants and the application of function symbols to other terms. Constraints in the latter domain involve equalities and disequalities between such terms, which may occur embedded in expressions together with several connectives and quantifiers, as reasoned before. This domain has been axiomatized and decision mechanisms have been described for the corresponding theory [15, 6]. However, since our approach can be considered as a *CLP* language, a wider domain is demanded, for which the tests of the satisfiability will be independent of the general goal solving procedure. The fields of application of *CLP* with constraint domains comprising the field of real numbers are abundant and very well known [10, 11]. In addition, thanks to the results due to Tarski about the decidability of the theory of real closed fields [19], this domain becomes a good choice as a constraint system instance. The present paper is an extended version of [7]. There we mixed the theory of real closed fields and the one of Herbrand terms, producing a constraint system called $\mathcal{RH}$ and the induced instance *HH($\mathcal{RH}$)* of the programming language based on *HH* with constraints. A partial algorithm to solve such $\mathcal{RH}$-constraints was also described. Now those notions are reviewed, performing slight, suitable modifications, and the presentation of the algorithms has been simplified and improved, as well as enriched with proofs that state their correctness. In addition, new results are also included, mainly a procedure responsible for the $\mathcal{RH}$-satisfiability decision of a subclass of constraints denominated *elemental constraints*.

There are other works in the literature which also introduce mixed numeric and symbolic domains (see e.g. [2, 5, 9]), but as far as we know all of them deal with the logic of Horn clauses, which implies that quantifiers in the goals are limited to be existential. Other approaches of decision procedures for combined theories, as [18], apply only to universal quantified formulas.

The structure of the rest of this paper is as follows. In Section 2, the constraint system $\mathcal{RH}$ is introduced in the context of the logic of hereditary Harrop formulas with constraints, and the expressive power of *HH*($\mathcal{RH}$) is illustrated. Section 3 is dedicated to present some preliminary definition and simple algorithms, useful to describe the procedure solving $\mathcal{RH}$-constraints. The first phase of this solver is explained in Section 4. The main results show up in two steps. In the first one, two rules that constitute

the core of a procedure to transform $\mathcal{RH}$-constraints into simpler ones are presented, together with the definitions and technical results needed to argue their foundations. In the second one, the algorithm that constitutes this procedure is detailed. Finally, its behavior is shown through examples. The second phase of the solver consists on the decision of the satisfiability of the output of the first phase. An algorithm to solve this problem for a subclass of $\mathcal{RH}$-constraints is carefully explained in Section 5. Finally, Section 6 contains a summary of the contributions of our work and states future research.

2 The Constraint System $\mathcal{RH}$

The logic of Hereditary Harrop formulas *HH* [16] arises when extending the logic of Horn clauses, on which the traditional logic programming language is based, relaxing the conditions over the structure of clauses and goals. Specifically, the connectives $\Rightarrow$ and $\forall$ are allowed in goals. A logic programming language called *HH(C)* is defined in [13], incorporating constraints of a generic constraint system $\mathcal{C}$ to *HH*. In it, the language and the conditions over a system $\mathcal{C}$ to be a valid choice as constraint system are specified, but no constraint solver for any particular system is studied. *HH(C)* may be briefly presented as the logic programming language whose clauses D and goals G are given by the following rules:

$$D ::= A \mid G \Rightarrow A \mid D_1 \wedge D_2 \mid \forall x D,$$

$$G ::= A \mid C \mid G_1 \wedge G_2 \mid G_1 \vee G_2 \mid D \Rightarrow G \mid C \Rightarrow G \mid \exists x G \mid \forall x G,$$

where A stands for atomic formulas, and C for constraints of the system $\mathcal{C}$.

The present paper focuses on a specific constraint system $\mathcal{RH} = \langle \mathcal{L}_{\mathcal{RH}}, \vdash_{\mathcal{RH}} \rangle$, which mixes $\mathcal{R}$eal numbers and finite symbolic trees (or $\mathcal{H}$erbrand terms). $\mathcal{L}_{\mathcal{RH}}$ denotes the set of formulas allowed as constraints, and $\vdash_{\mathcal{RH}}$ is the entailment relation that represents deducibility of a constraint C from a set of constraints Γ. $\mathcal{L}_{\mathcal{RH}}$ is a set of first-order formulas whose terms and atoms are built up using typed symbols belonging to a signature $\Sigma_{\mathcal{RH}}$. We agree on the existence of two sorts: r for real numbers, and h for Herbrand terms. Combining these basic types, types $\tau_1 \times \ldots \times \tau_n$ for predicate symbols and $\tau_1 \times \ldots \times \tau_n \rightarrow \tau$ for function symbols are built, where $\tau, \tau_i \in \{r, h\}$, $1 \leq i \leq n$. τ will stand for any type, and $o : \tau$ specifies that object o is of type τ. $\Sigma_{\mathcal{RH}}$ will contain:

- Constants, denoted by c. Some of them represent real numbers (-4.32, $1/2$, ...), and there is also a denumerable set of symbolic constants.
- Arithmetical operators $+, -, * : r \times r \rightarrow r$, and a finite or denumerable set of symbolic functions $f : \tau_1 \times \ldots \times \tau_n \rightarrow h$, where $\tau_i \in \{r, h\}, 1 \leq i \leq n$.
- The predicate symbol $\approx$: $\tau \times \tau$, where $\tau \in \{r, h\}$, for equality between terms of the same type.
- The symbols $<, >, \leq, \geq$: $r \times r$, for the classical order between real numbers.

Let $\mathcal{V} = \mathcal{V}_h \cup \mathcal{V}_r$ be the set of variables ν, where $\mathcal{V}_h$ is a denumerable set of variables of sort h, called symbolic variables, denoted with uppercase letters, and $\mathcal{V}_r$ is a denumerable set of variables of sort r, denoted with lowercase letters. The set of terms $\mathcal{T}_{\mathcal{RH}}$, with elements denoted t, is defined by the rule:

$$t ::= \nu \mid c \mid f(t_1, \ldots, t_n) \mid t_1 + t_2 \mid t_1 - t_2 \mid t_1 * t_2$$

where $f : \tau_1 \times \ldots \times \tau_n \to h$ and $t_i : \tau_i, 1 \leq i \leq n$. Notice that infix notation is used for arithmetical operators, and that the included terms t_1 and t_2 must be of type r.

The atoms of $\mathcal{L}_{\mathcal{RH}}$ are built up using the predicate symbols in $\Sigma_{\mathcal{RH}}$ (with infix notation) and the terms in $\mathcal{T}_{\mathcal{RH}}$. It also contains the constraints $\top$ and $\bot$, which stand for *true* and *false*, respectively. The existence of different connectives and quantifiers in goals forces the constraint system to be able to express and deduce complex constraints during the goal solving process, which specifically contain a mixed prefix of quantifiers. So, $\mathcal{L}_{\mathcal{RH}}$ must be enforced to be closed under $\wedge, \vee, \exists, \forall$ and $\neg$. For instance, if the signature contains the symbols $cons : r \times h \to h$ and $nil : h$ for the list constructor and the empty list, respectively, the formula

$$\forall x \exists X (X \approx cons(x * (2 - z), cons(-10.3, cons(z, nil))) \wedge \exists y (z \geq x + y))$$

is an $\mathcal{RH}$-constraint. The constraints with the form $t_1 \approx t_2$, where $t_1, t_2 : h$, will be referred to as *equalities*, and the ones with the form $\neg(t_1 \approx t_2)$ as *disequalities*. A *polynomial*, denoted by p, is a quantifier-free constraint such that its atomic subformulas are built up exclusively by means of predicate symbols and terms of type r. For instance, the constraint

$$(x \approx z * z - (x * (2.5 + (1/5) * y)) \vee z \geq x) \Rightarrow y * y > 4.24$$

is a polynomial. The notation $t(\overline{\nu})$ or $p(\overline{\nu})$ will specify that $\overline{\nu}$ is a set of variables comprising the ones occurring in t, or p, respectively. The substitution of a list of different variables $\overline{\nu}$ by a list of terms $\overline{t}$ of the same cardinality and corresponding types will be denoted by $[\overline{t}/\overline{\nu}]$, and its application to a term or formula, denoted $F[\overline{t}/\overline{\nu}]$, is defined in the usual way, avoiding the capture of free variables. $[\overline{t}/\overline{\nu}]$ is said to be a *symbolic ground substitution* if $\overline{t}$ contains no symbolic variables. It must be understood in the sequel that, given two lists of terms $\overline{s} \equiv s_1 \ldots s_n$ and $\overline{t} \equiv t_1 \ldots t_n$, where $n \geq 0$, $\overline{s} \approx \overline{t}$ stands for $s_1 \approx t_1 \wedge \ldots \wedge s_n \approx t_n$.

The deducibility relation between constraints, $\vdash_{\mathcal{RH}}$, is defined as the result of combining the classical deducibility relation with equality, $\vdash_{\approx}$, with the axiomatization $Ax_{\mathcal{R}}$ for the real closed fields due to Tarski [19], plus the axiomatization $Ax_{\mathcal{H}}$ for the Herbrand universe due to Clark [4]. I. e., given any set of constraints Γ and any constraint C, $\Gamma \vdash_{\mathcal{RH}} C$ if and only if $\Gamma \cup Ax_{\mathcal{R}} \cup Ax_{\mathcal{H}} \vdash_{\approx} C$. The relation $\vdash_{\mathcal{RH}}$ satisfies that $C_1 \vdash_{\mathcal{RH}} C_2$, if and only if, for any symbolic ground substitution σ, $C_1\sigma \vdash_{\mathcal{RH}} C_2\sigma$ holds.

C is said to be $\mathcal{RH}$-satisfiable if $\emptyset \vdash_{\mathcal{RH}} \overline{\exists} C$, where $\overline{\exists} C$ stands for the existential closure of C. We say that C and C' are $\mathcal{RH}$-equivalent, denoted by $C \equiv_{\mathcal{RH}} C'$, when $C \vdash_{\mathcal{RH}} C'$ and $C' \vdash_{\mathcal{RH}} C$. $\vdash_{\mathcal{RH}}$ satisfies that if $C_1 \equiv_{\mathcal{RH}} C_2$ and $C \vdash_{\mathcal{RH}} C'[C_1]$ (C_1 is a subformula of C'), then $C \vdash_{\mathcal{RH}} C'[C_2]$ (C_1 is replaced by C_2 in C').

Using this mixed constraint system as parameter, we obtain the instance *HH($\mathcal{RH}$)* of the logic programming language of hereditary Harrop formulas with constraints. As in *CLP*, the result of solving a goal using a program will be an *answer constraint*. Any goal solving procedure for *HH($\mathcal{RH}$)* (such as the general one defined in [13] for *HH($\mathcal{C}$)*) should incorporate a constraint solver capable of dealing with the satisfiability of partially or totally calculated answer constraints. Therefore, our goal has been to deal with the $\mathcal{RH}$-satisfiability problem for $\mathcal{RH}$-constraints. As far as we know, such task has not yet been proven decidable nor undecidable. It is important to notice that due to the fact that mixed quantifier prefixes may bind both real and symbolic

variables, no separate solvers dealing with Herbrand terms and reals, respectively, can be used. Specifically, it does not seem possible to take advantage directly of the mechanism defined in [18] for combining decision procedures for several theories into a single solver for their combination. The procedure presented in this paper is a partial solver, only suitable for a subclass of $\mathcal{RH}$-constraints.

Before going down into the details of the solving procedure, a simple example is shown in order to give evidence of the expressivity and syntactic components of $HH(\mathcal{RH})$.

Example 1 The $\mathcal{RH}$-constraints used in this example are polynomials, equalities or disequalities between lists of real numbers. The notation is that adopted in the language PROLOG, plus $\Rightarrow$ for implication and $\exists$, $\forall$ for existential and universal quantifiers in goals.

Let us consider a students database, and suppose that the students taking a specific course must solve several exercises, which are graded. The number of exercises is not fixed. In order to pass, any student should obtain marks between 0 and 10, and an average grade of at least 5.0. In addition, any student with marks higher than 9.0 in at least the half of the exercises gets a scholarship. The grades of some students are stored in the database. Other can be temporarily introduced in order to formulate hypothetical queries. One of the problems that students may tackle is to calculate the marks required in future exercises in order to get a scholarship.

The available information is gathered in the following program in *HH($\mathcal{RH}$)*. The grades obtained by the student named `juan` are stated by the first fact of this program. We assume standard PROLOG definitions for the predicates `sumelem(L,S)`, that given a list L succeeds if S is the sum of its elements, and `numelem(L,N)`, which succeeds if N is the number of elements of L.

```
grades(juan,[10,9.3,5]).
pass(X,L) :- grades(X,L), suitable(L), average(L,N), N>=5.
average(L,S) :- numelem(L,N), sumelem(L,N * S).
suitable([]).
suitable([X|L]) :- 0 <= X <= 10, suitable(L).
scholarship(X) :- pass(X,L),numelem(L,M),greatereq9(L,N),N>=(1/2)*M.
greatereq9([],0).
greatereq9([X|L],N+1) :- X >= 9, greatereq9(L,N).
greatereq9([X|L],N) :- X < 9, greatereq9(L,N).
convenient(X,L) :- grades(X,L) ⇒ scholarship(X).
```

The last clause allows to stablish hypothetical queries to the database and investigate the possible marks that a student should obtain in order to get a scholarship. When a goal `convenient(S, L)` must be solved, for a student S with a list L of grades, the fact `grades(S, L)` is temporarily introduced into the database during the solving of the goal.

Let us suppose that the student `pedro` wishes to know which mark he should try to get in the first two exercises, knowing that in the last his mark is 5, and he wants to get a scholarship, but not with the same grades as `juan`. This question may be expressed by the following goal, that includes a disequality between lists os reals.

$$\exists \texttt{L(grades(juan,L), convenient(pedro,[X,Y,5]), [X,Y,5]} \neq \texttt{L).}$$

The goal solver procedure should call a $\mathcal{RH}$-constraint solver in order to produce the expected answer constraint, namely:

$$(9 \leq X < 10 \wedge 9 \leq Y \leq 10) \vee (9 \leq X \leq 10 \wedge 9 \leq Y \leq 10 \wedge Y \neq 9.3).$$

Note that the variable L does not appear in the answer, because it is quantified in the goal.

Now, suppose that `pedro` wants to know if he may pass, knowing that his grade is the third part of `juan`'s grade in the second exercise and the half of `juan`'s grade in the first one. The answer to his dilemma should be *yes* and can be obtained by solving the existential goal:

```
∃A ∃B ∃L1 ∃L2(grades(juan,[2*A, 3*B |L1]), grades(pedro, [A,B|L2])
                        ⇒pass(pedro, [A,B|L2])).
```

The following sections are devoted to describe a solving procedure for $\mathcal{RH}$-constraints.

3 Classifying $\mathcal{RH}$-Constraints

The mechanism we will introduce to solve $\mathcal{RH}$-constraints is founded on a transformation of constraints into simpler $\mathcal{RH}$-equivalent ones, called *elemental constraints*, in which the real and symbolic parts are arranged in such a way that we are able to decide its $\mathcal{RH}$-satisfiability. Some steps in this simplification make use exclusively of properties of the connectives in first-order logic; others are based on the axiomatizations of the algebra of finite trees and real closed fields; the essential part is that dealing with quantifier elimination. The transformation mechanism combines the techniques due to Maher [15] for symbolic quantified variables, with the elimination of quantifiers over real variables using of *C*ylindrical *A*lgebraic *D*ecompositions (*CAD*) [3].

Several definitions are needed prior to the presentation of the procedure.

3.1 Solved Forms, Basic Formulas and Elemental Constraints

In order to define the concept of elemental constraints, whose $\mathcal{RH}$-satisfiability we are able to decide, we previously introduce some notions regarding some constraints that play a special role during the description of the procedure presented later.

Definition 1 *A* system of equalities *E is a finite conjunction of equalities. A* system of equalities and polynomial *E is a conjunction of a system of equalities and one polynomial.*

Definition 2 *E is said to be a* solved form *if $E \equiv \overline{X} \approx \overline{t}(\overline{U}, \overline{x}) \wedge p(\overline{x})$, where each variable in $\overline{X}$ appears exactly once. The variables in $\overline{U}$ will be referred to as* parameters *of the system, and the ones in $\overline{X}$ as its* eliminable variables*. If $X \approx t$ appears in E and V occurs in t, we say that X* depends on *V. Given two systems of equalities and polynomial, E_1 and E_2, E_2 is said to be a* solved form of *E_1 if $E_1 \equiv_{\mathcal{RH}} E_2$ and E_2 is a solved form.*

Definition 3 *A* basic formula *is a constraint of the form $\exists \overline{U} E$, where E is a system of equalities in solved form and $\overline{U}$ its set of parameters. The constraint $\top$ is also regarded as basic.*

The set of boolean combinations of basic formulas *is defined as the least set that contains every basic formula and is closed under* $\vee$, $\wedge$ *and* $\neg$.

In the sequel, basic formulas will be denoted by b, possibly with sub or superscripts.

Definition 4 *An* $\mathcal{RH}$-normal form *is a constraint of the form* $C_1 \vee \ldots \vee C_n$, $n \geq 1$, *where* $C_i \equiv b^i \wedge \neg b^i_1 \wedge \ldots \wedge \neg b^i_k \wedge p^i$, $k \geq 0$, $1 \leq i \leq n$. *A* quantified $\mathcal{RH}$-normal form *is a constraint of the form* ΠC, *where* C *is an* $\mathcal{RH}$*-normal form and* Π *is a sequence of quantifications. An* elemental constraint *is a quantified* $\mathcal{RH}$*-normal form* $\Pi_1\Pi_2 C$, *in which* Π_1 *is a sequence of existential quantifications and* Π_2 *binds only real variables.*

In the search process for $\mathcal{RH}$-equivalent elemental constraints, there are two ways of eliminating real quantifiers, one of them simplifies constraints containing polynomials following the definition below.

Definition 5 *A* solved polynomial *on* x *is a polynomial of the form* $(x \approx t_1 \wedge p_1) \vee \ldots \vee (x \approx t_l \wedge p_l)$, *where* x *does not appear in* $t_1, \ldots, t_l, p_1, \ldots, p_l$.

The idea behind this definition is that the variable x can be represented as a function of the others, by means of a finite number of equations that can be seen as real substitutions. Then it is allowed to replace the existential quantifier over x by the disjunction of the application of the different substitutions.

3.2 The Algorithm Solve-Tree

In this subsection an algorithm is described whose purpose is to find a solved form of a system of equalities and polynomial. It is based on the standard rules for equality solving in the algebra of finite trees [8]. The algorithm, designated by SOLVE-TREE, simply consists on the sequential, nondeterministic application of the rules below, until none of them can be applied. Such rules are written in the form $\frac{E_1}{E_2}$, which means that the system of equalities and polynomial E_1 may be transformed by this rule into the system E_2.

SOLVE-TREE:

1) $\frac{f(t_1,\ldots,t_n)\approx f(t'_1,\ldots,t'_n)\wedge E}{t_1\approx t'_1\wedge\ldots\wedge t_n\approx t'_n\wedge E}$ for any symbolic function symbol f.

2) $\frac{f(t_1,\ldots,t_n)\approx g(t'_1,\ldots,t'_m)\wedge E}{\bot}$ if f and g are different symbolic function symbols.

3) $\frac{c\approx c'\wedge E}{\bot}$ if c and c' are different symbolic constants.

4) $\frac{X\approx X\wedge E}{E}$

5) $\frac{t\approx X\wedge E}{X\approx t\wedge E}$ if t is not a variable.

6) $\frac{X\approx t\wedge E}{X\approx t\wedge E[t/X]}$ if X does not occur in t, and X occurs in some equality in E.

7) $\frac{X\approx t\wedge E}{\bot}$, if t is not a variable and X occurs in t.

In the rule 1), the expressions $t_i \approx t'_i$, $t_i, t'_i : r$, obtained by decomposition, are conjunctively added to the polynomial of E.

Properties of SOLVE-TREE**:**

- Every rule in SOLVE-TREE is sound, in the sense that, if it transforms a system E_1 into E_2, then both systems are $\mathcal{RH}$-equivalent.
- For any E used as input, SOLVE-TREE terminates, producing a solved form for E if there is any, or $\bot$ otherwise. In the latter case, we say that it *fails*.
- A system of equalities and polynomial $\overline{X} \approx \overline{t} \wedge p$ is a solved form if and only if no rule can be applied to it. It is $\mathcal{RH}$-satisfiable if and only if it has a solved form $\overline{X} \approx \overline{t} \wedge p$ and p is $\mathcal{RH}$-satisfiable.
- SOLVE-TREE provides for a solved form E' of a system E, if there is any. However, E may have other solved forms, characterized as follows. Let $\Theta(E')$ be the set of variable renames θ with domain and range exactly the set of symbolic variables free in E, and such as, if $V\theta = V' \neq V$, then $V \approx V'$ or $V' \approx V$ is in E'. Then, for every $\theta \in \Theta(E')$, $E'\theta$ is another solved form of E. Conversely, for every other solved form E'' of E, there exists $\theta \in \Theta(E')$ such that $E'\theta$ has the same eliminable variables that E''. The elements of $\Theta(E')$ can be regarded as permutations, in the classical algebraic sense, of the set of symbolic variables free in E. Particularly, if $\theta \in \Theta(E')$ is the permutation of exactly two variables V_1 and V_2, then we say that $E'\theta$ is obtained by *swapping* the equality $V_1 \approx V_2$ of E', i.e., if $E' \equiv V_1 \approx V_2 \wedge E''$ the swapping which interchanges V_1 and V_2 transforms E' into $V_2 \approx V_1 \wedge E''[V_1/V_2]$.

3.3 The Algorithm for $\mathcal{RH}$-Normalization

We are able to decide the $\mathcal{RH}$-satisfiability of elemental constraints (see Subsection 5). Therefore, the first aim of the $\mathcal{RH}$-constraint solver will be to reach elemental constraints by eliminating quantifiers of quantified $\mathcal{RH}$-normal forms. During the process, it will often be necessary to transform boolean combinations of basic formulas and polynomials into $\mathcal{RH}$-equivalent normal forms. Such normalization can be easily accomplished using the algorithm described below.

NORMALIZE:

1) Distribute every negation, until the scope of all of them is a unique basic formula. Transform the result into a disjunction of constraints of the form $b_1 \wedge \ldots \wedge b_n \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_m \wedge p$ under a quantifier prefix.
2) Each constraint $C \equiv b_1 \wedge \ldots \wedge b_n \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_m \wedge p$ in such disjunction with $n > 1$ must be replaced by $b \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_m \wedge (p \wedge p')$, where b and p' are obtained as follows:
   2.1) Let $b_i \equiv \exists \overline{U}^i (\overline{X}_i \approx \overline{t}_i)$, $1 \leq i \leq n$, and let $E \wedge p'$ be the solved form of $\overline{X}_1 \approx \overline{t}_1 \wedge \ldots \wedge \overline{X}_n \approx \overline{t}_n$, produced by SOLVE-TREE, if it does not fail. Otherwise, C is simply removed from the original disjunction.
   2.2) Let $\overline{U} = \overline{U}^1 \cup \ldots \cup \overline{U}^n$. Let E' be the result of eliminating in E all the equalities concerning the eliminable variables of E belonging to $\overline{U}$, and let $\overline{W}$ be the set of variables of $\overline{U}$ not appearing in E'. Then, $b \equiv \exists \overline{U} \backslash \overline{W} E'$.

   In the special case in which all the constraints C in the disjunction were removed, the output is defined as $\neg\top$, that is the negation of a basic formula, hence a $\mathcal{RH}$-normal form.

4 Eliminating Quantifiers

The procedure that transforms a $\mathcal{RH}$-constraint into a simpler one proceeds performing quantifier elimination in quantified $\mathcal{RH}$-normal forms. The goal of that procedure is to find an elemental constraint $\mathcal{RH}$-equivalent to the initial one. Nevertheless, some conditions on the form of the constraint are needed in order to guarantee that the elimination of a quantifier could be performed. We have found a condition, with respect to the type and order on the occurrence of the real and symbolic quantifications. In addition, a real quantifier may be removed in some other cases for which that condition is not fulfilled. This is the case when the included polynomials are equivalent to others that are solved on the variable whose quantifier will be eliminated. Linear polynomials are an example of such case. See for instance the works included in [1], that show techniques capable to find such solved forms.

Now the transformation algorithm as well as its theoretical foundations will be presented.

4.1 Theoretical Foundations of the Transformation Algorithm

Quantifiers over real variables are eliminated using algebraic methods as *CAD*, [3]. The elimination of symbolic quantifiers is based on two rules, (*Pair*) and (*Elim*), which transform constraints into $\mathcal{RH}$-equivalent simpler ones that are boolean combinations of basic formulas. Such rules will be presented, together with the theorems that constitute the foundations of their soundness. The theorem upon which the first rule relies is similar to that presented in [15] for symbolic terms because, in some sense, the polynomial has been isolated. Nevertheless, we have elaborated an original proof, based on the entailment relation $\vdash_{\mathcal{RH}}$ instead of on the semantics, as it was done in [15], since that specific semantics is not suitable for $\mathcal{RH}$-constraints because the presence of real terms inside symbolic functions. The following lemma is required in such proof.

Lemma 1 *Let t be a term of type h, with free symbolic variables $\overline{X}$, and let Γ be a set of $\mathcal{RH}$-constraints, with no free symbolic variables in $\overline{X}$. Let t_1, t_2 be terms with no free symbolic variables, and such that t_2 results from t_1, replacing one or more symbolic subterms by new terms whose principal function symbol, or constant, does not appear in t, t_1, Γ. Then, if there is no substitution σ with domain $\overline{X}$ such that $\Gamma \vdash_{\mathcal{RH}} t\sigma \approx t_1$, there is also no substitution θ with domain $\overline{X}$ such that $\Gamma \vdash_{\mathcal{RH}} t\theta \approx t_2$.*

Theorem 2 (Pair) *Let $b, b_1, \ldots, b_n$ be basic formulas, $n \geq 1$. Then,*

$$F_1 \equiv \exists Y(b \wedge \neg b_1 \wedge \ldots \wedge \neg b_n) \equiv_{\mathcal{RH}} \exists Y(b \wedge \neg b_1) \wedge \ldots \wedge \exists Y(b \wedge \neg b_n) \equiv F_2.$$

Proof: Without loss of generality, we can suppose that the sets of parameters in the basic formulas $b \wedge b_1 \wedge \ldots \wedge b_n$ are disjoint, not containing occurrences of the variable Y. It can also be supposed that Y appears as eliminable in all of them, since, for any basic formula $b^\star$ not containing occurrences of Y, $b^\star \equiv \exists Y'(b^\star \wedge Y \approx Y')$, where Y' is a variable not occurring in $b^\star$. Thereafter, $b^\star$ may be replaced by $\exists Y'(b^\star \wedge Y \approx Y')$, and conversely. Let us write $b \equiv \exists \overline{U}, \overline{W}\, E$, where $E \equiv \overline{X} \approx \overline{t}(\overline{U}, \overline{x}) \wedge Y \approx s(\overline{V}, \overline{W}, \overline{x})$. $\overline{U}$ is the set of symbolic variables in $\overline{t}$. $\overline{x}$ is the set of variables of sort r in E. The set of symbolic variables in the term s is divided in two, $\overline{V}$ and $\overline{W}$, depending on whether variables belong to $\overline{U}$ or not, respectively. Thus, $\overline{V} \subseteq \overline{U}$, and $\overline{W} \cap \overline{U} = \emptyset$. For each $1 \leq i \leq n$, we write b_i in an analogous way, adding the superscripts i to every set of symbolic variables. Let $\overline{X}^\star = (\overline{X} \cup \overline{X}^1 \cup \ldots \cup \overline{X}^n)/Y$.

$F_1 \vdash_{\mathcal{RH}} F_2$ is trivial. We will prove $F_2 \vdash_{\mathcal{RH}} F_1$. Let σ be a symbolic ground substitution with domain $\overline{X}^\star$ and let $\sigma^\star$ be a symbolic ground substitution, extending σ to $\overline{U}, \overline{W}, Y$, and such that for $\overline{U}$ it is defined in such a way that $F_2\sigma \vdash_{\mathcal{RH}} (\overline{X} \approx \overline{t})\sigma^\star$ holds; for each $W \in \overline{W}$, let $W\sigma^\star$ be a term whose principal function symbol, or constant, does not appear in $F_1\sigma$, $F_2\sigma$. Finally, let $Y\sigma^\star = s\sigma^\star$. Our goal is to prove that $F_2\sigma \vdash_{\mathcal{RH}} (E \wedge \neg b_1 \wedge \ldots \wedge \neg b_n)\sigma^\star$. Once this is proved, $F_2\sigma \vdash_{\mathcal{RH}} F_1\sigma$ will be straightforward, and $F_2 \vdash_{\mathcal{RH}} F_1$ is followed by the properties of $\mathcal{RH}$.

Due to the definition of $\sigma^\star$, $F_2\sigma \vdash_{\mathcal{RH}} E\sigma^\star$ holds trivially. Therefore, it only remains to provide a proof for

$$F_2\sigma \vdash_{\mathcal{RH}} \neg b_i\sigma^\star,\ 1 \leq i \leq n \qquad (\dagger).$$

Let i, $1 \leq i \leq n$, then $F_2\sigma \vdash_{\mathcal{RH}} (\exists Y(b \wedge \neg b_i))\sigma$ holds. Hence, the entailment relation $\vdash_{\mathcal{RH}}$ satisfies that there is a ground symbolic substitution σ_i, extending σ to $\overline{U}, \overline{W}, Y$ such that:

$$F_2\sigma \vdash_{\mathcal{RH}} (E \wedge \neg b_i)\sigma_i \qquad (\ddagger).$$

Let σ_i' be a substitution that extends σ_i to $\overline{U}^i$ for which $F_2\sigma \vdash_{\mathcal{RH}} (\overline{X}^i \approx \overline{t}^i)\sigma_i'$, if there is any. Otherwise, $F_2\sigma \vdash_{\mathcal{RH}} (\neg\overline{X}^i \approx \overline{t}^i)\sigma'$ for every substitution σ' extending σ, and $(\dagger)$ is straightforward. But if such σ_i' exists, since $(\ddagger)$ implies $F_2\sigma \vdash_{\mathcal{RH}} \neg b_i\sigma_i$, there can be no further extension σ_i'' of it to $\overline{W}^i$ such that

$$F_2\sigma \vdash_{\mathcal{RH}} Y\sigma_i \approx s^i(\overline{V}^i\sigma_i', \overline{W}^i\sigma_i'', \overline{x}) \qquad (\sharp).$$

Now, in order to prove $(\dagger)$, we will show that there is no extension θ of $\sigma^\star$ to $\overline{U}^i, \overline{W}^i$ such that $F_2\sigma \vdash_{\mathcal{RH}} E_i\theta$. Let us suppose that there is an extension θ' of $\sigma^\star$ to $\overline{U}^i$ such that $F_2\sigma \vdash_{\mathcal{RH}} (\overline{X}^i \approx \overline{t}^i)\theta'$. We conclude if we show that there is no such θ extending θ' to $\overline{W}^i$, for which $F_2\sigma \vdash_{\mathcal{RH}} Y\sigma^\star \approx s^i(\overline{V}^i\theta', \overline{W}^i\theta, \overline{x})$ holds.

From $(\ddagger)$ we deduce that $F_2\sigma \vdash_{\mathcal{RH}} (\overline{X} \approx \overline{t})\sigma_i$, but $\sigma^\star$ is a substitution for $\overline{U}$ satisfying $F_2\sigma \vdash_{\mathcal{RH}} (\overline{X} \approx \overline{t})\sigma^\star$, thus $F_2\sigma \vdash_{\mathcal{RH}} \overline{U}\sigma_i \approx \overline{U}\sigma^\star$ holds. On the other hand σ_i' is a substitution for $\overline{U}^i$ satisfying $F_2\sigma \vdash_{\mathcal{RH}} (\overline{X}^i \approx \overline{t}^i)\sigma_i'$, and we are supposing $F_2\sigma \vdash_{\mathcal{RH}} (\overline{X}^i \approx \overline{t}^i)\theta'$, hence $F_2\sigma \vdash_{\mathcal{RH}} \overline{U}^i\sigma_i' \approx \overline{U}^i\theta'$. Therefore, it will be enough to prove that there is no such extension θ for which $F_2\sigma \vdash_{\mathcal{RH}} Y\sigma^\star \approx s^i(\overline{V}^i\sigma_i', \overline{W}^i\theta, \overline{x})$ holds. From $(\ddagger)$ we obtain $F_2\sigma \vdash_{\mathcal{RH}} Y\sigma_i \approx s(\overline{V}\sigma_i, \overline{W}\sigma_i, \overline{x})$, and using $(\sharp)$ we deduce that there is no substitution σ_i'' extending σ_i' to $\overline{W}^i$ such that $F_2\sigma \vdash_{\mathcal{RH}} s(\overline{V}\sigma_i, \overline{W}\sigma_i, \overline{x}) \approx s^i(\overline{V}^i\sigma_i', \overline{W}^i\sigma_i'', \overline{x})$. Hence, since $s(\overline{V}\sigma^\star, \overline{W}\sigma^\star, \overline{x})$ can be obtained from $s(\overline{V}\sigma_i, \overline{W}\sigma_i, \overline{x})$ by replacing some subterms by terms whose principal symbol function, or constant, does not appear in the previous formulas, Lemma 1 can be applied, and states that there is no extension θ of θ' to $\overline{W}^i$ satisfying $F_2\sigma \vdash_{\mathcal{RH}} s(\overline{V}\sigma^\star, \overline{W}\sigma^\star, \overline{x}) \approx s^i(\overline{V}^i\sigma_i', \overline{W}^i\theta, \overline{x})$, concluding the proof since $F_2\sigma \vdash_{\mathcal{RH}} \overline{V}^i\sigma_i' \approx \overline{V}^i\theta'$, and $F_2\sigma \vdash_{\mathcal{RH}} s(\overline{V}\sigma^\star, \overline{W}\sigma^\star, \overline{x}) \approx Y\sigma^\star$, by definition of $\sigma^\star$. ■

We are finally ready to define the rule $(Pair)$, whose purpose is to eliminate conjunctions of negated basic formulas under the same existential quantifier.

Definition 6 *The rule $(Pair)$ replaces a formula of the form $\exists Y(b \wedge \neg b_1 \wedge \ldots \wedge \neg b_n)$ by the $\mathcal{RH}$-equivalent formula provided by Theorem 2.*

Now, we present several definitions and technical results useful to justify the rule (*Elim*), which is the responsible for the elimination of symbolic quantifiers, together with a proof of its correctness in Theorem 11.

Definition 7 *Given two sets of symbolic variables $\overline{V}$ and $\overline{W}$, and a system of equalities and polynomial E, we say that E* constrains *$\overline{V}$* w.r.t. *$\overline{W}$ if and only if, for every solved form of E, either some $V \in \overline{V}$ is eliminable, or some $W \in \overline{W}$ is eliminable and depends on some $V \in \overline{V}$. Otherwise, we say that E* does not constrain *$\overline{V}$* w.r.t. *$\overline{W}$.*

It is straightforward to check that if E_1 and E_2 are two solved forms such that $E_1 \equiv_{\mathcal{RH}} E_2$, then E_1 constrains $\overline{V}$ w.r.t. $\overline{W}$ if and only if E_2 constrains $\overline{V}$ w.r.t. $\overline{W}$. This definition involves to check a property for every solved form of E, which is difficult to check directly. However, it is possible to provide for a characterization of this notion very easy to use. Some technical lemmas must be previously introduced.

Lemma 3 *Let $E(\overline{V}, \overline{W}, \overline{Z}, \overline{x})$ be a system of equalities and polynomial in solved form. Then, E does not have the form $\overline{W}_e \approx \overline{t}(\overline{W}_p, \overline{Z}_p, \overline{x}) \wedge \overline{Z}_e \approx \overline{s}(\overline{W}_p, \overline{Z}_p, \overline{V}, \overline{x}) \wedge p(\overline{x})$, if and only if, at least one of the conditions below holds:*

1) *E contains an equality $W \approx t$, where $W \in \overline{W}$ and some variable in $\overline{V}$ occurs in t.*
2) *E contains an equality $V \approx t$, where $V \in \overline{V}$ and t is not a variable.*
3) *E contains an equality $V \approx X$, where $V \in \overline{V}, X \in \overline{V} \cup \overline{W}$.*
4) *E contains an equality $V \approx Z$, where $V \in \overline{V}$, $Z \in \overline{Z}$, and the parameter Z appears also in an equality $X \approx t$, where $X \in \overline{V} \cup \overline{W}$.*
5) *E contains an equality $V \approx Z$, where $V \in \overline{V}$, $Z \in \overline{Z}$, and the parameter Z does not appear in any equality $X \approx t$, where $X \in \overline{V} \cup \overline{W}$.*

Proof: If E is in solved form, it has the form $\overline{W}_e \approx \overline{t}(\overline{W}_p, \overline{Z}_p, \overline{x}) \wedge \overline{Z}_e \approx \overline{s}(\overline{W}_p, \overline{Z}_p, \overline{V}, \overline{x}) \wedge p(\overline{x})$ iff (a) no variable $V \in \overline{V}$ is eliminable and (b) no variable $W \in \overline{W}$ depends on any $V \in \overline{V}$. (a) holds iff $2), 3), 4)$ and 5) do not hold. (b) holds iff 1) does not hold. ■

Lemma 4 *Let $E(\overline{V}, \overline{W}, \overline{Z}, \overline{x})$ be a system of equalities and polynomials in solved form. If E constrains $\overline{V}$ w.r.t $\overline{W}$, then at least one of the conditions 1) to 4) of Lemma 3 holds for E.*

Proof: By reductio ad absurdum. Let us suppose that none of the conditions 1) to 4) holds for E. Now, we must face two different cases:

a) Condition 5) does not hold for E. Then, by virtue of Lemma 3, E has the form $\overline{W}_e \approx \overline{t}(\overline{W}_p, \overline{Z}_p, \overline{x}) \wedge \overline{Z}_e \approx \overline{s}(\overline{W}_p, \overline{Z}_p, \overline{V}, \overline{x}) \wedge p(\overline{x})$, therefore E does not constrain $\overline{V}$ w.r.t $\overline{W}$.

b) Condition 5) holds for E. Then, swapping all the equalities in E of the form $V \approx Z$, where $V \in \overline{V}, Z \in \overline{Z}$, a new solved form E' is produced, for which conditions 1) to 5) do not hold. Thus, E' does not constrain $\overline{V}$ w.r.t $\overline{W}$ and, since $E \equiv_{\mathcal{RH}} E'$, neither does E. ■

Lemma 5 *Let E be a system of equalities, and $E_1(\overline{V}, \overline{W}, \overline{Z}, \overline{x})$, $E_2(\overline{V}, \overline{W}, \overline{Z}, \overline{x})$ two solved forms of E. Then, one of the conditions 1) to 4) in Lemma 3 holds for E_1, if and only if one of the conditions 1) to 4) holds for E_2.*

Proof: Given two $\mathcal{RH}$-equivalent solved forms with the same set of eliminable variables, any of the conditions 1) to 5) holds for one of them if and only if it holds for the other. Let us suppose that one of the conditions 1) to 4) holds for E_1, and let $\theta \in \Theta(E_1)$ be the permutation for which $E_1\theta$ has the same eliminable variables that E_2. Notice that any $\theta \in \Theta(E_1)$ can be seen as the composition of a finite number of swaps. Therefore it suffices to prove that if E_1' is obtained by swapping E_1, then one of the conditions 1) to 4) holds for E_1 if and only if one of the conditions 1) to 4) holds for E_1'. In the present proof we only analyze those swaps that change the equalities which cause that some of the conditions 1) to 4) holds for E_1, since for the other swaps the same condition still holds for E_1', obviously.

- E_1 satisfies 1), i.e., E_1 contains an equality $W \approx t$, where $W \in \overline{W}$ and some variable in $\overline{V}$ occurs in t. Relevant swaps:
  - If $t \equiv V$ and E_1' is obtained by swapping $W \approx t$. Then $V \approx W$ is in E_1', and thus condition 3) holds for E_1'.
  - E_1' is obtained by swapping $X \approx V$, for some variable X. Then, E_1' contains the equalities $V \approx X$ and $W \approx t[X/V]$. Again, different cases may arise, depending on the nature of X.
    - If $X \in \overline{V} \cup \overline{W}$, condition 3) holds for E_1'.
    - If $X \in \overline{Z}$, condition 4) holds for E_1'.
- E_1 satisfies 2), i.e., E_1 contains an equality $V \approx t$, where $V \in \overline{V}$ and t is not a variable. Then, trivially, any E_1' obtained by swapping any equality will still contain an equality with this form, and therefore condition 2) will hold for E_1'.
- E_1 satisfies 3), i.e., E_1 contains an equality $V \approx Y$, where $V \in \overline{V}, Y \in \overline{V} \cup \overline{W}$. Relevant swaps:
  - E_1' is obtained by swapping $V \approx Y$. Then, if $Y \in \overline{V}$ condition 3) still holds, otherwise condition 1) holds.
  - E_1' is obtained by swapping one equality of the form $X \approx Y$, for some variable X. Then, it contains the equalities $V \approx X$ and $Y \approx X$. Possible cases:
    - If $X \in \overline{V} \cup \overline{W}$, condition 3) holds for E_1'.
    - If $X \in \overline{Z}$, condition 4) holds for E_1'.
- E_1 satisfies 4), therefore, E_1 contains the equalities $V \approx Z, X \approx t$, where $V \in \overline{V}$, $Z \in \overline{Z}$, and some variable in $\overline{V} \cup \overline{W}$ appears in t. There are basically two relevant swaps.
  - E_1' is obtained by swapping the equality $V \approx Z$. Therefore, it contains the equalities $Z \approx V$ and $X \approx t[V/Z]$. Possible cases:
    - If $X \in \overline{V}$, conditions 2) or 3) hold for E_1'.
    - If $X \in \overline{W}$, condition 1) holds for E_1'.

– E_1' is obtained by swapping one equality of the form $Y \approx Z$, for some variable Y. Then, it contains the equalities $V \approx Y$ and $X \approx t[Y/Z]$, and hence condition 4) also holds for E_1'. ■

Now, we are ready to prove the syntactic characterization that decides whether a solved form constrains a set of variables with respect to another. It will be required in the proof of Theorem 11 (Elim).

Proposition 6 (Characterization) *Let $E(\overline{V}, \overline{W}, \overline{Z}, \overline{x})$ be a solved form. E constrains $\overline{V}$ w.r.t. $\overline{W}$ if and only if one of the conditions below holds for E.*

1) E contains $W \approx t$, where $W \in \overline{W}$ and some variable in $\overline{V}$ occurs in t.

2) E contains $V \approx t$, where $V \in \overline{V}$ and t is not a variable.

3) E contains $V \approx X$, where $V \in \overline{V}, X \in \overline{V} \cup \overline{W}$.

4) E contains $V \approx Z$ and $X \approx t$, where $V \in \overline{V}$, $Z \in \overline{Z}$, $X \in \overline{V} \cup \overline{W}$ and X depends on Z.

Proof: One direction is proved just re-stating Lemma 4. For the other, let us assume that one of the conditions 1) to 4) holds for E. Then, by virtue of Lemma 5, the same happens to any other solved form of E, and therefore, thanks to Lemma 3, no solved form of E can be written in the form $\overline{W}_e \approx \overline{t}(\overline{W}_p, \overline{Z}_p, \overline{x}) \wedge \overline{Z}_e \approx \overline{s}(\overline{W}_p, \overline{Z}_p, \overline{V}, \overline{x}) \wedge p(\overline{x})$, i.e. E constrains $\overline{V}$ w.r.t $\overline{W}$. ■

Lemma 7 *Let $\overline{X}_1, \overline{X}_2$ and $\overline{U}$ be disjoint sets of symbolic variables, and let $\overline{V}$ be such that $\overline{V} \subseteq \overline{U}$. Then, for any set $\overline{x}$ of real variables and vectors of terms $\overline{t}_1, \overline{t}_2$, the equivalence below holds:*

$$\exists \overline{X}_1 \exists \overline{V}(\overline{X}_1 \approx \overline{t}_1(\overline{U}, \overline{x}) \wedge \overline{X}_2 \approx \overline{t}_2(\overline{U}, \overline{x}) \wedge p(\overline{x})) \equiv_{\mathcal{RH}} \exists \overline{V}(\overline{X}_2 \approx \overline{t}_2(\overline{U}, \overline{x}) \wedge p(\overline{x})).$$

Proof: (sketch) This result is a consequence of the fact that $\vdash_{\mathcal{RH}}$ is founded on $\vdash_{\approx}$, the classical logic deduction with equality. Specifically, it follows from the more general equivalence between formulas presented below. Notice that it is presented in the context of unsorted classical logic, therefore all variables are denoted with lowercase letters.

Let $F_1 \equiv \exists \overline{x} \exists \overline{y}(x_1 \approx t_1 \wedge \ldots \wedge x_n \approx t_n \wedge F)$ and $F_2 \equiv \exists \overline{y} F$, where

- F_1, F_2, F are formulas in any signature.
- $\overline{x}$ and $\overline{y}$ are disjoint sets of variables.
- $t_1, \ldots t_n$ are any terms in which the variables $x \in \overline{x}$ do not occur.
- F is any formula in which the variables $x \in \overline{x}$ do not occur.

Then, it is straightforward to check that F_1 and F_2 are equivalent according to $\vdash_{\approx}$. ■

Lemma 8 *Let F_1 be an equality system with set of parameters $\overline{U}_1$, and let $\exists \overline{U}_2 F_2$ be another constraint where F_2 is quantifier-free. Then, the following equality holds:*

$$\exists \overline{U}_1(F_1 \wedge \neg \exists \overline{U}_2 F_2) \equiv_{\mathcal{RH}} \exists \overline{U}_1 F_1 \wedge \neg \exists \overline{U}_1, \overline{U}_2(F_1 \wedge F_2).$$

Proof: (sketch) The claim the latter lemma is forced by the axioms for finite trees, together with the properties of $\vdash_{\approx}$. Specifically, it follows from the more general equivalence between formulas presented below, again in the context of unsorted classical logic.

Let $F_1 \equiv \exists\overline{y}(x_1 \approx t_1 \wedge \ldots \wedge x_n \approx t_n \wedge \neg\exists\overline{z}F)$ and $F_2 \equiv \exists\overline{y}(x_1 \approx t_1 \wedge \ldots \wedge x_n \approx t_n) \wedge \neg\exists\overline{y}\exists\overline{z}F$, where

- F_1, F_2, F are formulas in any signature.
- $\overline{x}$ and $\overline{y}$ and $\overline{z}$ are disjoint sets of variables.
- $t_1, \ldots t_n$ are any terms in which only variables $y \in \overline{y}$ occur.
- F is any formula.

Then, it is easy to check that $Ax_{\mathcal{H}}, F_1 \vdash_{\approx} F_2$ and $Ax_{\mathcal{H}}, F_2 \vdash_{\approx} F_1$ hold. ■

The following lemma is an important one, that justifies by itself the notion of systems of equalities and polynomial constraining a set of variables with respect to another. It states that, under certain conditions, an $\mathcal{RH}$-equivalent constraint is obtained when replacing an initial prefix of existential quantifiers by a universal one. Intuitively, this allows us to replace $\neg\exists\overline{V}\exists\overline{Z}F$ by $\neg\forall\overline{V}\exists\overline{Z}F$ and therefore by $\exists\overline{V}\neg\exists\overline{Z}F$. The proof of Theorem 11 (Elim) will take advantage of this fact.

Lemma 9 *For any system of equalities $E(\overline{V}, \overline{W}, \overline{Z}, \overline{x})$, the following statements hold:*

a) If E does not constrain $\overline{V}$ w.r.t $\overline{W}$, then $\exists\overline{V}\exists\overline{Z}\,E \equiv_{\mathcal{RH}} \forall\overline{V}\exists\overline{Z}\,E$.

b) Otherwise, $\forall\overline{V}\exists\overline{Z}\,E \equiv_{\mathcal{RH}} \bot$.

Proof: Let $F_1 \equiv \exists\overline{V}\exists\overline{Z}\,E$ and $F_2 \equiv \forall\overline{V}\exists\overline{Z}\,E$.

a) Let us suppose that E does not constrain $\overline{V}$ w.r.t $\overline{W}$, so there exists a solved form $E' \equiv \overline{Z}_e \approx \overline{t}(\overline{Z}_p, \overline{W}_p, \overline{V}, \overline{x}) \wedge \overline{W}_e \approx \overline{s}(\overline{Z}_p, \overline{W}_p, \overline{x}) \wedge p(\overline{x})$ of E. We now prove the $\mathcal{RH}$-equivalence between F_1 and F_2. $F_2 \vdash_{\mathcal{RH}} F_1$ is straightforward. For $F_1 \vdash_{\mathcal{RH}} F_2$, it is enough to prove that, for any ground symbolic substitution σ, $F_1\sigma \vdash_{\mathcal{RH}} F_2\sigma$ holds. Since E and E' are $\mathcal{RH}$-equivalent, $F_1\sigma \vdash_{\mathcal{RH}} \exists\overline{V}\exists\overline{Z}\,E'\sigma$ also holds, thus there is a substitution σ^* that extends σ to $\overline{Z}_p$, such that $F_1\sigma \vdash_{\mathcal{RH}} \exists\overline{V}\exists\overline{Z}_e\,E'\sigma^*$. In fact for every substitution θ extending σ^* to $\overline{V}$, $F_1\sigma \vdash_{\mathcal{RH}} \exists\overline{Z}_e\,E'\theta$, because the variables in $\overline{V}$ are parameters only in equalities of the form $\overline{Z}_e = \overline{t}$. That implies $F_1\sigma \vdash_{\mathcal{RH}} \forall\overline{V}\exists\overline{Z}_e\,E'\sigma^*$. Thus, we conclude $F_1\sigma \vdash_{\mathcal{RH}} \exists\overline{Z}_p\forall\overline{V}\exists\overline{Z}_e\,E'\sigma$, and therefore $F_1\sigma \vdash_{\mathcal{RH}} \forall\overline{V}\exists\overline{Z}\,E\sigma$, by the properties of $\vdash_{\mathcal{RH}}$ and the $\mathcal{RH}$-equivalence between E and E'.

b) Let us suppose that E constrains $\overline{V}$ w.r.t $\overline{W}$. Now, if E has no solved form, then it is not $\mathcal{RH}$-satisfiable, hence $F_2 \equiv_{\mathcal{RH}} \bot$. Otherwise, let E' be a solved form of E. In virtue of Proposition 6, one of the conditions 1)–4) in Lemma 3 holds for E'. In the following, we analyze all cases.

1) E' contains an equality $W \approx t$, where $W \in \overline{W}$ and some variable in $\overline{V}$ occurs in t. Obviously, if t contains only the parameters $\overline{V}' \subseteq \overline{V}$, then $\emptyset \vdash_{\mathcal{RH}} \exists\overline{V}'(W \not\approx t)$. This fact can be generalized, as follows. Let $\overline{V}' \cup \overline{Z}'$ be the set of parameters that appear in t, where $\overline{Z}' \subseteq \overline{Z}$ and $\overline{V}' \subseteq \overline{V}$. Then, is easy to check that $\emptyset \vdash_{\mathcal{RH}} \exists\overline{V}'\forall\overline{Z}'(W \not\approx t)$. Hence, $\forall\overline{V}'\exists\overline{Z}'(W \approx t) \equiv_{\mathcal{RH}} \bot$. Therefore, $F_2 \equiv_{\mathcal{RH}} \bot$, because $F_2 \vdash_{\mathcal{RH}} \forall\overline{V}'\exists\overline{Z}'(W \approx t)$.

2) E' contains an equality $V \approx t$, where $V \in \overline{V}$ and t is not a variable. Let us assume that $\overline{V}'$ and $\overline{Z}'$ are defined as in the previous case. Then, clearly $\emptyset \vdash_{\mathcal{RH}} \exists V, \overline{V}' \forall \overline{Z}' (V \not\approx t)$, therefore $\forall V, \overline{V}' \exists \overline{Z}' (V \approx t) \equiv_{\mathcal{RH}} \bot$. Again, this implies $F_2 \equiv_{\mathcal{RH}} \bot$.

3) If E' contains an equality $V_1 \approx V_2$, where $V_1, V_2 \in \overline{V}$, then, since $\forall \overline{V}(V_1 \approx V_2) \equiv_{\mathcal{RH}} \bot$, we can again conclude $F_2 \equiv_{\mathcal{RH}} \bot$. Otherwise, if E' contains an equality $V \approx W$, where $V \in \overline{V}$ and $W \in \overline{W}$, since $\emptyset \vdash_{\mathcal{RH}} \exists V (V \not\approx W)$, $\forall V (V \approx W) \equiv_{\mathcal{RH}} \bot$ holds, and thereafter the same conclusion is reached.

4) E' contains an equality $V \approx Z$, where $V \in \overline{V}$, $Z \in \overline{Z}$, and the parameter Z appears also in some other equality for some variable in $\overline{V} \cup \overline{W}$. We must deal with two cases.

   i) Z appears as parameter in another equality for $\overline{V}$, i.e, there is another equality $V_2 \approx t$, and Z appears in t. If $t \not\equiv Z$, also condition 2) holds, and the proof is concluded. Otherwise, $\forall V V_2 \exists Z (V \approx Z \wedge V_2 \approx Z) \equiv_{\mathcal{RH}} \bot$ holds, and thereafter $F_2 \equiv_{\mathcal{RH}} \bot$.

   ii) Z appears as parameter in another equality for $\overline{W}$, i.e, there is another equality $W \approx t$, and Z appears in t. $\emptyset \vdash_{\mathcal{RH}} \exists V (\neg (V \approx Z \wedge W \approx t))$, then $\forall V (V \approx Z \wedge W \approx t) \equiv_{\mathcal{RH}} \bot$ holds, so $F_2 \equiv_{\mathcal{RH}} \bot$.

Thus, in all cases, $F_2 \equiv_{\mathcal{RH}} \bot$ is concluded. ■

The following corollary, which is an immediate consequence of the previous lemma, is useful to deal with formulas in the form $\exists X \neg b$, where b is a basic formula. These are not boolean combinations of basic formulas but, according to the corollary, an equivalent boolean combination of basic formulas can be easily found.

Corollary 10 *Let $E(\overline{V}, \overline{W}, \overline{Z}, \overline{x})$ be a system of equalities, and $F \equiv \exists \overline{V} \neg \exists \overline{Z}\, E$. If E constrains $\overline{V}$ w.r.t $\overline{W}$, then $F \equiv_{\mathcal{RH}} \top$. Otherwise, $F \equiv_{\mathcal{RH}} \neg \exists \overline{V} \exists \overline{Z}\, E$.*

Proof: It is an immediate consequence of Lemma 9 and the $\mathcal{RH}$-equivalence between $\exists \overline{U} \neg \exists \overline{V}\, E$ and $\neg \forall \overline{U} \exists \overline{V}\, E$, for any system of equalities E. ■

Finally, once all the previous machinery has been duly developed, it is possible to provide a proof for the theorem which is directly responsible for the symbolic quantifier elimination, namely:

Theorem 11 (Elim) *Let b and b' be basic formulas. Let Y be a variable that may appear in them only as eliminable variable. Then, the formula $F \equiv \exists Y (b \wedge \neg b')$ is $\mathcal{RH}$-equivalent to one of the following constraints, depending on the case:*

1. *Y does not appear in b nor in b'. Then, trivially $F \equiv_{\mathcal{RH}} b \wedge \neg b'$.*
2. *Y appears in b, but not in b'. Let us write $b \equiv \exists \overline{U} (\overline{X} \approx \overline{t}(\overline{U}, \overline{x}) \wedge Y \approx s(\overline{U}, \overline{x}))$. Then, $F \equiv_{\mathcal{RH}} \exists \overline{U} (\overline{X} \approx \overline{t}(\overline{U}, \overline{x})) \wedge \neg b'$.*
3. *Y appears in b', but not in b. Let us write $b' \equiv \exists \overline{U} (\overline{X} \approx \overline{t}(\overline{U}, \overline{x}) \wedge Y \approx s(\overline{U}, \overline{x}))$. If $\overline{X} \approx \overline{t}(\overline{U}, \overline{x}) \wedge Y \approx s(\overline{U}, \overline{x})$ constrains Y w.r.t. $\overline{X}$, then $F \equiv_{\mathcal{RH}} b$. Otherwise, $F \equiv_{\mathcal{RH}} b \wedge \neg \exists \overline{U} (\overline{X} \approx \overline{t}(\overline{U}, \overline{x}))$.*
4. *Y appears in both b and b', written $b \equiv \exists \overline{U}, \overline{V} (\overline{X} \approx \overline{t}(\overline{U}, \overline{x}) \wedge Y \approx s(\overline{U}, \overline{V}, \overline{x}))$ and $b' \equiv \exists \overline{U}^1 (\overline{X}^1 \approx \overline{t}^1(\overline{U}^1, \overline{x}) \wedge Y \approx s^1(\overline{U}^1, \overline{x}))$.*

a) *If* $\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge \overline{X}^1 \approx \overline{t}^1(\overline{U}^1,\overline{x}) \wedge s(\overline{U},\overline{V},\overline{x}) \approx s^1(\overline{U}^1,\overline{x})$ *constrains* $\overline{V}$ *w.r.t.* $\overline{X}$. *Then* $F \equiv_{\mathcal{RH}} \exists\overline{U}(\overline{X} \approx \overline{t}(\overline{U},\overline{x}))$.

b) *If* $\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge \overline{X}^1 \approx \overline{t}^1(\overline{U}^1,\overline{x}) \wedge s(\overline{U},\overline{V},\overline{x}) \approx s^1(\overline{U}^1,\overline{x})$ *does not constrain* $\overline{V}$ *w.r.t.* $\overline{X}$. *Then, if the algorithm* SOLVE-TREE *fails when processing* $\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge \overline{X}^1 \approx \overline{t}^1(\overline{U}^1,\overline{x}) \wedge s(\overline{U},\overline{V},\overline{x}) \approx s^1(\overline{U}^1,\overline{x})$, $F \equiv_{\mathcal{RH}} \exists\overline{U}(\overline{X} \approx \overline{t}(\overline{U},\overline{x}))$. *Otherwise,* $F \equiv_{\mathcal{RH}} \exists\overline{U}(\overline{X} \approx \overline{t}(\overline{U},\overline{x})) \wedge \neg\exists\overline{V}\,\overline{U}\,\overline{U}^1 \backslash \overline{W} E$, *where* E *is obtained by removing in the result produced by* SOLVE-TREE *all the equalities corresponding to the eliminable variables belonging to* $\overline{V} \cup \overline{U} \cup \overline{U}^1$, *and* $\overline{W}$ *is the subset of* $\overline{V} \cup \overline{U} \cup \overline{U}^1$ *of the variables not occurring in* E.

Proof: The previous four cases must be considered.

1. Y does not appear in b nor in b'. Then, trivially $F \equiv_{\mathcal{RH}} b \wedge \neg b'$.
2. Y appears in b, but not in b'. Thanks to Lemma 7, $F \equiv_{\mathcal{RH}} \exists\overline{U}(\overline{X} \approx \overline{t}(\overline{U},\overline{x})) \wedge \neg b'$.
3. Y appears in b', but not in b. Obviously, $F \equiv_{\mathcal{RH}} b \wedge \exists Y \neg b'$. The second formula in the conjunction can be simplified using Corollary 10, as follows. If $\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge Y \approx s(\overline{U},\overline{x})$ constrains Y w.r.t $\overline{X}$, then $F \equiv_{\mathcal{RH}} b$. Otherwise, $F \equiv_{\mathcal{RH}} b \wedge \neg\exists\overline{U}(\overline{X} \approx \overline{t}(\overline{U},\overline{x}))$.
4. Y appears in both b and b'. First, we state that $\exists Y(b \wedge \neg b') \equiv_{\mathcal{RH}} \exists\overline{U},\overline{V}(\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge \neg\exists\overline{U}^1(\overline{X}^1 \approx \overline{t}^1(\overline{U}^1,\overline{x}) \wedge s(\overline{U},\overline{V},\overline{x}) \approx s^1(\overline{U}^1,\overline{x})))$. Using Lemma 8 this is $\mathcal{RH}$-equivalent to $\exists\overline{V}(\exists\overline{U}(\overline{X} \approx \overline{t}(\overline{U},\overline{x})) \wedge \neg\exists\overline{U},\overline{U}^1(\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge \overline{X}^1 \approx \overline{t}^1(\overline{U}^1,\overline{x}) \wedge s(\overline{U},\overline{V},\overline{x}) \approx s^1(\overline{U}^1,\overline{x})))$. $\overline{V}$ does not appear in the first formula of the conjunction, hence the whole formula can be simplified to $\exists\overline{U}(\overline{X} \approx \overline{t}(\overline{U},\overline{x})) \wedge \exists\overline{V}\neg\exists\overline{U},\overline{U}^1(\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge \overline{X}^1 \approx \overline{t}^1(\overline{U}^1,\overline{x}) \wedge s(\overline{U},\overline{V},\overline{x}) \approx s^1(\overline{U}^1,\overline{x}))$. The second formula in this conjunction can be transformed by the application of Corollary 10, in the following way. If $\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge \overline{X}^1 \approx \overline{t}^1(\overline{U}^1,\overline{x}) \wedge s(\overline{U},\overline{V},\overline{x}) \approx s^1(\overline{U}^1,\overline{x})$ does not constrain $\overline{V}$ w.r.t $\overline{X}$, then $F \equiv_{\mathcal{RH}} \exists\overline{U}(\overline{X} \approx \overline{t}(\overline{U},\overline{x})) \wedge \neg\exists\overline{V},\overline{U},\overline{U}^1(\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge \overline{X}^1 \approx \overline{t}^1(\overline{U}^1,\overline{x}) \wedge s(\overline{U},\overline{V},\overline{x}) \approx s^1(\overline{U}^1,\overline{x}))$ (otherwise, $F \equiv_{\mathcal{RH}} \exists\overline{U}(\overline{X} \approx \overline{t}(\overline{U},\overline{x}))$, and the proof is concluded). This formula is not yet a boolean combination of basic formulas and polynomials, but it can be easily transformed into an $\mathcal{RH}$-equivalent one. This is performed by using the algorithm SOLVE-TREE for the set of equalities $\overline{X} \approx \overline{t}(\overline{U},\overline{x}) \wedge \overline{X}^1 \approx \overline{t}^1(\overline{U}^1,\overline{x}) \wedge s(\overline{U},\overline{V},\overline{x}) \approx s^1(\overline{U}^1,\overline{x})$. If we denote by E' the resulting solved form, then in virtue of Lemma 7, $\exists\overline{V},\overline{U},\overline{U}^1 E'$ is simplified to the constraint $\exists(\overline{V},\overline{U},\overline{U}^1 \backslash \overline{W})E$ in the heading of the theorem. ■

Definition 8 *The rule* (Elim) *replaces a formula in the form* $\exists Y(b \wedge \neg b')$ *by the* $\mathcal{RH}$*-equivalent boolean combination of basic formulas with a polynomial provided by Theorem 11.*

Notice that the resulting polynomial is a conjunction of real equalities that is obtained only in the case 4.b) of Theorem 11. The rule (*Elim*) will be also applied when b' does not exist, since that can be considered as a particular case of 2).

4.2 The Transformation Algorithm

As we have mentioned before, the algorithm that checks $\mathcal{RH}$-satisfiability should be classified as a quantifier elimination technique. Such elimination can be performed only in some cases, depending on the order of the symbolic and real quantifications, and under some conditions of solvability over the included polynomials, which will be shown during the description of the algorithm. The process requires a prior manipulation that converts the initial constraint into a quantified $\mathcal{RH}$-normal form, for which the elimination quantifier techniques can be applied. In the following, we detail the whole algorithm. Its input is an arbitrary constraint C, its output is an $\mathcal{RH}$-equivalent elemental constraint.

Phase I: Preprocessing.
`Input`: An arbitrary constraint C
`Output`: A constraint $\mathcal{RH}$-equivalent to C in quantified $\mathcal{RH}$-normal form.
`Procedure`: The sequential applications of the following steps should be carried out.

1) Produce a constraint in prenex form $\mathcal{RH}$-equivalent to C. In this way, a formula $\Pi C'$ is obtained so that $C \equiv_{\mathcal{RH}} \Pi C'$, where Π is a sequence of quantifiers and C' is quantifier-free.

2) Transform $\Pi C'$ into $\Pi(C_1 \vee \ldots \vee C_n)$, where, for each $1 \leq i \leq n$, C_i is a conjunction of equalities, disequalities and polynomials.

3) Transform $\Pi(C_1 \vee \ldots \vee C_n)$ into an $\mathcal{RH}$-equivalent quantified $\mathcal{RH}$-normal form, modifying each C_i, $1 \leq i \leq n$, as follows.

   3.1) If the system of equalities and polynomial in C_i has no solved form E, remove C_i from the disjunction. Otherwise, replace it by $p \wedge b$, defined as follows. Let $\overline{W}$ be the set of parameters of E and let $\overline{W}'$ be a set of fresh variables in bijection with $\overline{W}$. Let $p \wedge b$ be the result of moving the polynomials outside the scope of the quantification $\exists \overline{W'}$ in $\exists \overline{W}' (E[\overline{W} / \overline{W}'])$. If E boils down to a polynomial, take $b \equiv \top$.

   3.2) For each disequality in C_i remove the outermost negation and proceed as in step 3.1). Add conjunctively the result, preceded by the negation symbol, to $p \wedge b$.

   3.3) At this stage, C_i has been removed or transformed into $p \wedge b \wedge \neg(p_1 \wedge b_1) \wedge \ldots \wedge \neg(p_s \wedge b_s)$, $s \geq 0$, which is finally $\mathcal{RH}$-normalized.

Phase II: Quantifier elimination algorithm.
The goal of this stage is to decrease the length of the quantifier sequence Π as much as possible, replacing $\Pi C''$ by another $\mathcal{RH}$-equivalent constraint whose quantifier prefix is shorter than Π. We can choose the length of their sequence of quantifiers as a complexity measure for quantifier $\mathcal{RH}$-normal forms, and therefore such goal may be viewed as a simplification algorithm, which we will refer to as SIMPLIFY. Essentially, it works repeatedly applying another procedure named ELIMINATE, which eliminates the innermost quantifier of a given quantifier $\mathcal{RH}$-normal form. However, such elimination can only be performed under certain conditions, that will be previously checked. These algorithms are presented below.

SIMPLIFY: It transforms a quantified $\mathcal{RH}$-normal form into a simpler constraint.
`Input`: Any quantified $\mathcal{RH}$-normal form $\Pi C \equiv \Pi(D_1 \vee \ldots \vee D_m)$, where $D_i \equiv p^i \wedge b^i \wedge \neg b^i_1 \wedge \ldots \wedge \neg b^i_{k_i}$, $1 \leq i \leq m$.
`Output`: A simpler $\mathcal{RH}$-equivalent constraint.

$\Pi^\star :=$ REORD(Π);
`while`$(\Pi^\star \neq \emptyset)$ `do`
  $\Pi' Q\nu :=$ ELEMENT$(\Pi^\star)$;
  $C' :=$ ELIMINATE$(Q\nu(D_1 \vee \ldots \vee D_m))$;
  `if` $(C' \neq \bot)$
    `then`
      $C :=$ NORMALIZE(C');
      $\Pi := \Pi'$;
      $\Pi^\star :=$ REORD(Π)
    `else` $\Pi^\star := \Pi^\star - \{\Pi' Q\nu\}$
  `end if`
`end while`;
`return` ΠC.

Where if $\Pi = \Pi_1\Pi_2$, and $\Pi_2 = Q\nu_1 \ldots Q\nu_k$ is the maximal existential (or universal) suffix of Π, then REORD$(\Pi) = \{\Pi_1 Q\nu_1\, Q\nu_2 \ldots Q\nu_{1-i}\, Q\nu_{1+i} \ldots Q\nu_k\, Q\nu_i, 1 \leq i \leq k\}$. ELEMENT selects an element from a set.

The termination of the transformation algorithm SIMPLIFY is guaranteed, because the number of elements of REORD(Π) decreases in every loop.

ELIMINATE: It eliminates $Q\nu$ in $Q\nu(D_1 \vee \ldots \vee D_m)$, when it is possible.
`Input`: $Q\nu C \equiv Q\nu(D_1 \vee \ldots \vee D_m)$. Each D_i with the form $p^i \wedge b^i \wedge \neg b^i_1 \wedge \ldots \wedge \neg b^i_{k_i}$.
`Output`: An $\mathcal{RH}$-equivalent boolean combination of basic formulas, or $\bot$ if $Q\nu$ is not eliminable.

1: `if` $(Q = \exists)$ `then` $C := \exists\nu\, D_1 \vee \ldots \vee \exists\nu\, D_m$;
  1.1: `if` $(\nu = x : r)$ `then` $i := 1$; $B := true$;
  `while` $(i \leq m$ `and` $B)$ `do`
    1.1(a): `if` (x occurs in p^i `and` x does not occur in $b^i_1, \ldots, b^i_{k_i}$)
      `then` Replace in C, $\exists x D_i$ by $p^{i\prime} \wedge b^i \wedge \neg b^i_1 \wedge \ldots \wedge \neg b^i_{k_i}$, where $p^{i\prime}$ is a quantifier-free Tarski formula $\mathcal{RH}$-equivalent to $\exists x\, p^i$, obtained by applying a *CAD*-based algorithm to it
    `end if` 1.1(a);
    1.1(b): `if` (x occurs in $b_i \wedge \neg b'_{i1} \wedge \ldots \wedge \neg b'_{ik}$, `and` a polynomial of the form $(x \approx t^i_1 \wedge q^i_1) \vee \ldots \vee (x \approx t^i_{l_i} \wedge q^i_{l_i})$, solved for x, $\mathcal{RH}$-equivalent to p^i, can be found)
      `then` Replace in C, $\exists x D_i$ by the $\mathcal{RH}$-equivalent constraint
        $(q^i_1 \wedge (b^i \wedge \neg b^i_1 \wedge \ldots \wedge \neg b^i_{k_i})[t^i_1/x]) \vee \ldots \vee$
        $(q^i_{l_i} \wedge (b_i \wedge \neg b^i_1 \wedge \ldots \wedge \neg b^i_{k_i})[t^i_{l_i}/x])$
    `end if` 1.1(b);
    1.1(c): `if` (x occurs in $b_i \wedge \neg b'_{i1} \wedge \ldots \wedge \neg b'_{ik}$, `and` a polynomial of the form $(x \approx t^i_1 \wedge q^i_1) \vee \ldots \vee (x \approx t^i_{l_i} \wedge q^i_{l_i})$, solved for x, $\mathcal{RH}$-equivalent to p^i, is not found)
      `then` $B := false$
    `end if` 1.1(c);
    i := i +1

```
      end while;
      if B then return C else return ⊥ end if
   end if 1.1;
   1.2: if (ν = X : h) then
      for i := 1 to m do  D_i' := b^i ∧ ¬b^i_1 ∧ ... ∧ ¬b^i_{k_i};
         Replace ∃X D_i in C by the RH-equivalent constraint p^i ∧ ∃X D'_i;
         if (k_i = 0)
            then Transform ∃X D'_i into a basic formula
            else Apply the rule (Pair), replacing ∃X D'_i by
               ∃X (b^i ∧ ¬b^i_i) ∧ ... ∧ ∃X (b^i ∧ ¬b^i_{k_i});
               Transform each member of this conjunction,
               by means of the rule (Elim),
               into a boolean combination of polynomials and basic formulas
         end if;
      end for;
      return C
   end if 1.2
end if 1;
2: if (Q = ∀) then
   return ¬ ELIMINATE (∃ν NORMALIZE (¬(D_1 ∨ ... ∨ D_m)))
end if 2
```

Note that the conditions to eliminate an existential quantifier are determined by the steps 1.1(a), 1.1(b) and 1.2 of ELIMINATE. Universal quantified constraints are considered in 2 and are transformed into the negation of an existential quantified one, prior to the elimination. The condition of 1.1(b) can be determined for certain polynomials using techniques based on Gröebner bases [1]. For these polynomials, the wanted solved form can be built from intermediate steps of the quantifier elimination *CAD* methods.

We propose a partial constraint solver for the CLP programming language $HH(\mathcal{RH})$, based on the transformation algorithm just described. The first phase of the solving procedure consists on the application of the algorithm SIMPLIFY to a given constraint. The next phase should be to decide the $\mathcal{RH}$-satisfiability of the output of the transformation. If such output is an elemental constraint, then the solver will proceed with the algorithm ELEMENTAL-SAT described in Section 5. If not, this solver cannot decide the satisfiability of the initial constraint.

4.3 Examples

The aim of this subsection is to provide examples of the usage of the simplification algorithm. Examples of programs in *HH(RH)*, such as the one introduced in Section 2, can be found in [13], together with the description of the goal solving procedure. The latter needs to make use of the constraint solver described in the present paper, in order to check $\mathcal{RH}$-satisfiability of answer constraints.

Example 2 Let C_1 be the constraint below, where $f : r \to h$, $g : h \times h \to h$.

$$C_1 \equiv \exists X(\exists x, y(g(X,Y) \approx g(f(x), f(y)) \wedge x + y \approx 1) \wedge \forall Z\, \neg(g(X,Y) \approx g(Z,Z))).$$

The algorithm would transform C_1 as follows. The preprocessing phase yields:

$$\exists X \exists x, y \forall Z \Big((X \approx f(y) \wedge Y \approx f(x)) \wedge \neg \exists Z_1 (X \approx Z_1 \wedge Y \approx Z_1 \wedge Z \approx Z_1) \wedge x + y \approx 1 \Big).$$

SIMPLIFY is used now. In the first loop, $Q\nu = \forall Z$ is chosen, and ELIMINATE.2 transforms the quantification $\forall Z$ is transformed into $\exists Z$, and then $\mathcal{RH}$-normalization is carried out, rendering:

$$\exists X \exists x, y \neg \exists Z \Big(\neg (X \approx f(y) \wedge Y \approx f(x)) \vee \exists Z_1 (X \approx Z_1 \wedge Y \approx Z_1 \wedge Z \approx Z_1) \vee \neg (x + y \approx 1) \Big),$$

and ELIMINATE is recursively called with $\exists \nu = \exists z$. According ELIMINATE.1 $\exists Z$ must be distributed, the subconstraint $\exists Z, Z_1 (X \approx Z_1 \wedge Y \approx Z_1 \wedge Z \approx Z_1)$ is simplified by ELIMINATE.1.2 into $\exists Z_1 (X \approx Z_1 \wedge Y \approx Z_1)$. Notice that $(k = 0)$.

The $\mathcal{RH}$-normalization in the loop SIMPLIFY yields:

$$\exists X \exists x \exists y \Big((X \approx f(y) \wedge Y \approx f(x)) \wedge \neg \exists Z_1 (X \approx Z_1 \wedge Y \approx Z_1) \wedge x + y \approx 1 \Big).$$

In the next loop $Q\nu = \exists X$ is chosen, and step ELIMINATE.1.2 is applied. The rule $(Elim)$ produces:

$$\exists x \exists y ((Y \approx f(x)) \wedge \neg (Y \approx f(x) \wedge x \approx y) \wedge x + y \approx 1).$$

This transformation comprises several steps, immediately detailed. First, the constraint

$$\exists x \exists y \Big((Y \approx f(x)) \wedge \neg \exists Z_1 (Y \approx f(x) \wedge Y \approx Z_1 \wedge f(y) \approx Z_1) \wedge x + y \approx 1 \Big)$$

is obtained, which does not already contain quantifiers over X. Then, SOLVE-TREE must be applied to $Y \approx f(x) \wedge Y \approx Z_1 \wedge f(y) \approx Z_1$. The result is simplified eliminating the equality for Z_1 and its quantifier.

Now, NORMALIZE produces the quantified $\mathcal{RH}$-normal form below:

$$\exists x \exists y \Big((x + y \approx 1 \wedge Y \approx f(x) \wedge \neg Y \approx f(x)) \vee (x + y \approx 1 \wedge Y \approx f(x) \wedge \neg x \approx y) \Big).$$

Following ELIMINATE.1, it is transformed into:

$$\exists x \Big(\exists y (x + y \approx 1 \wedge Y \approx f(x) \wedge \neg Y \approx f(x)) \vee \exists y (x + y \approx 1 \wedge Y \approx f(x) \wedge \neg x \approx y) \Big).$$

For the first constraint in the disjunction, ELIMINATE.1.1(a) can be now applied, transforming $\exists y (x + y \approx 1)$ into $\top$. The second one, following ELIMINATE.1.1(a) again, $\exists y (x + y \approx 1 \wedge \neg (x \approx y))$ is transformed into $\neg (x \approx 1/2)$. Therefore,

$$C_1 \equiv_{\mathcal{RH}} \exists x \Big((Y \approx f(x) \wedge \neg Y \approx f(x)) \vee (Y \approx f(x) \wedge \neg (x \approx 1/2)) \Big).$$

At this stage ELEMENTAL-SAT is used in order to the decide the $\mathcal{RH}$-satisfiability of the latter (elemental) constraint, which turns out to be $\mathcal{RH}$-satisfiable.

Example 3 In this example, we will deal with symbolic terms built up using the list constructor to handle lists of reals. For the sake of readability, the usual PROLOG notation for lists is adopted. Let us begin with the constraint C_2:

$$\exists L(\exists x_2, L_2(L \approx [x_1, x_2|L_2] \wedge x_1 + x_2 \approx 4) \wedge \neg\exists x_3, L_3(L \approx [x_3|L_3] \wedge x_3 \times x_3 \approx 1)).$$

The preprocessing phase yields:

$$\exists x_2 \exists L \forall x_3(x_1 + x_2 \approx 4 \wedge \exists L_2(L \approx [x_1, x_2|L_2]) \wedge \neg(x_3 \times x_3 \approx 1 \wedge \exists L_3(L \approx [x_3|L_3])).$$

The negation was not distributed because the next step will be to transform the quantifier $\forall$ into the corresponding $\exists$, and this would imply to undo this distribution.

SIMPLIFY is applied. In the first loop, $Q\nu = \forall x_3$ is the only possible choice, and according to ELIMINATE.2, after the $\mathcal{RH}$-normalization and the distribution of the existential quantification $\exists x_3$ the constraint obtained is:

$$\begin{array}{c}\exists x_2 \exists L \neg(\exists x_3(x_1 + x_2 \not\approx 4) \vee \exists x_3(\neg\exists L_2(L \approx [x_1, x_2|L_2]))\vee \\ \exists x_3(x_3 \times x_3 \approx 1 \wedge \exists L_3(L \approx [x_3|L_3]))).\end{array}$$

The quantification $\exists x_3$ is trivially removed in the first two cases by ELIMINATE1.1(a). For the last one, the polynomial $x_3 \times x_3 \approx 1$ can be transformed into $x_3 \approx 1 \vee x_3 \approx -1$, which is solved for x. Therefore, step ELIMINATE.1.1(b), followed by the execution of NORMALIZE, renders:

$$\begin{array}{c}\exists x_2 \exists L(x_1 + x_2 \approx 4 \wedge \exists L_2(L \approx [x_1, x_2|L_2])\wedge \\ \neg\exists L_3(L \approx [1|L_3]) \wedge \neg\exists L_3(L \approx [-1|L_3])).\end{array}$$

At this stage, a new loop of SIMPLIFY is carried out, taking $Q\nu = \exists L$. Step ELIMINATE.1.2 is considered, and the application of the rule $(Pair)$ yields:

$$\begin{array}{c}\exists x_2(x_1 + x_2 \approx 4 \wedge \exists L(\exists L_2(L \approx [x_1, x_2|L_2]) \wedge \neg\exists L_3(L \approx [1|L_3]))\wedge \\ \exists L(\exists L_2(L \approx [x_1, x_2|L_2]) \wedge \neg\exists L_3(L \approx [-1|L_3]))).\end{array}$$

The rule $(Elim)$ must be applied to

$$\exists L(\exists L_2(L \approx [x_1, x_2|L_2]) \wedge \neg\exists L_3(L \approx [1|L_3])),$$

producing $\neg x_1 \approx 1$. This constraint is obtained applying SOLVE-TREE to $[x_1, x_2|L_2] \approx [1|L_3]$, and simplifying trivial equalities and quantifiers. Analogously, applying $(Elim)$ to

$$\exists L(\exists L_2(L \approx [x_1, x_2|L_2]) \wedge \neg\exists L_3(L \approx [-1|L_3])),$$

it is reduced to $\neg x_1 \approx -1$. Thus the constraint produced in this step is

$$\exists x_2(x_1 + x_2 \approx 4 \wedge \neg x_1 \approx 1 \wedge \neg x_1 \approx -1).$$

The last loop of SIMPLIFY is carried out now, using ELIMINATE.1.1(a), the current constraint is transformed into:

$$\neg x_1 \approx 1 \wedge \neg x_1 \approx -1.$$

The procedure ELEMENTAL-SAT, described in Section 5, can be used now to prove that the latter constraint is in fact $\mathcal{RH}$-satisfiable, and therefore so is C_2.

Example 4 This example shows a simple subclass of quantifier $\mathcal{RH}$-normal constraints for which the procedure SIMPLIFY is useless. Let us consider the constraint

$$C_3 \equiv \forall X \exists x[\exists U(X \approx f(U, x)) \wedge s(x) \geq 0],$$

where $s(x)$ is any term of type r in which only the variable x occurs. The procedure ELIMINATE cannot perform the elimination of the quantifier $\exists x$, since x occurs in the basic formula $\exists U(X \approx f(U,x))$ and the polynomial $s(x) \geq 0$ is not $\mathcal{RH}$-equivalent to any polynomial with the form $x \approx t_1 \vee \ldots \vee x \approx t_n$. However, it is easy to check that $C_3 \equiv_{\mathcal{RH}} \bot$ holds for any $s(x)$.

We can see here that the order among quantifiers plays a critical role. If $\exists x$ and $\forall X$ are swapped, then SIMPLIFY promptly returns the expected result $\bot$.

5 Satisfiability of Elemental Constraints

The purpose of this section is the description of a $\mathcal{RH}$-satisfiability decision procedure for elemental constraints, which is called ELEMENTAL-SAT. It should be regarded as the second phase of the solving procedure for $\mathcal{RH}$-constraints. Two versions of ELEMENTAL-SAT are presented. The first one is able to deal with a particularly simple subclass of elemental constraints. The second one is an enhancement that can be used with any elemental constraint. The section begins with the theoretical foundations for such procedures, and it ends with an example of the usage and limitations of ELEMENTAL-SAT.

5.1 Theoretical Foundations of the Procedure *Elemental-Sat*

Let us begin introducing some definitions and lemmas that will be useful to prove the soundness of ELEMENTAL-SAT.

Definition 9 *We define the relation $\equiv_h$ between terms as the least one for which the conditions below hold.*

1. *$t \equiv_h t$ for any symbolic term $t : h$.*
2. *$t_1 \equiv_h t_2$ for any real terms $t_1 : r, t_2 : r$.*
3. *$f(t_1, \ldots, t_n) \equiv_h f(s_1, \ldots, s_n)$ iff $t_i \equiv_h s_i$ for every i, $1 \leq i \leq n$.*

If $t_1 \equiv_h t_2$ we say that t_1 is equal on its symbolic part *to t_2. Trivially, $\equiv_h$ is an equivalence relation.*

Definition 10 *A substitution σ is a* symbolic unifier *for the system of equalities $s_1 \approx t_1 \wedge \ldots \wedge s_n \approx t_n$ if $s_i\sigma \equiv_h t_i\sigma$, for every i, $1 \leq i \leq n$. A symbolic unifier is said to be a* non-arithmetical-operators *unifier (*n.a.o–unifier*) if no arithmetical operators nor real constants occur in its range. A n.a.o–unifier σ is* most general *if for any other n.a.o–unifier σ', there is a substitution τ such that $t\sigma\tau \equiv_h t\sigma'$ for any symbolic term $t : h$.*

Trivially, if σ is a symbolic unifier for E and σ' results from replacing in σ any real subterm in its range by any other, σ' is also a symbolic unifier for E.

The soundness of the algorithm ELEMENTAL-SAT, which will be described later on, is based on the following results and technicalities.

The algorithm works with quantified $\mathcal{RH}$-normal forms such that every basic formula occurring in them has the same set of eliminable variables. If a constraint C in $\mathcal{RH}$-normal form does not satisfy the previous condition, its *completion* is computed as follows:

- Let $\overline{X}$ be the union of the sets of eliminable variables of the basic formulas occurring in C.
- Rename the parameters in all basic formulas occurring in C with fresh variable names, taking care that the renaming is also performed in the quantifier prefix.
- Later, replace the basic formulas b by $\exists U(b \wedge X \approx U)$, where U is a new fresh parameter for b and $X \in \overline{X}$ is not eliminable for b, until all basic formulas share a common set of eliminable variables.

Lemma 12 *The completion of a quantified $\mathcal{RH}$-normal form C is $\mathcal{RH}$-equivalent to C.*

This lemma follows from the fact that the renaming of parameters and the replacement of basic formulas b by expressions described in the previous paragraph preserve the $\mathcal{RH}$-equivalence.

Lemma 13 *Let $C \equiv \overline{X} \approx \overline{t} \wedge \neg\exists\overline{V}(\overline{X} \approx \overline{s})$, where the variables in $\overline{X}$ are all distinct not occurring in $\overline{s}$ nor in $\overline{t}$, the sets of variables in $\overline{s}$ and $\overline{t}$ are disjoint, and $\exists\overline{V}(\overline{X} \approx \overline{s})$ is a basic formula. Then:*

1. *If E is the result of applying* SOLVE-TREE *to $\overline{s} \approx \overline{t}$, then $C \equiv_{\mathcal{RH}} \overline{X} \approx \overline{t} \wedge \neg\exists\overline{V}E$.*
2. *If* SOLVE-TREE *fails in the previous case, then $C \equiv_{\mathcal{RH}} \overline{X} \approx \overline{t}$.*

Proof: $C \equiv \overline{X} \approx \overline{t} \wedge \neg\exists\overline{V}(\overline{X} \approx \overline{s})$ is equivalent to $\overline{X} \approx \overline{t} \wedge \neg\exists\overline{V}(\overline{t} \approx \overline{s})$ in the classical logic with equality, so they are $\mathcal{RH}$-equivalent. Then, since SOLVE-TREE preserves $\mathcal{RH}$-equivalence, they are $\mathcal{RH}$-equivalent to $\overline{X} \approx \overline{t} \wedge \neg\exists\overline{V}E$. In case that SOLVE-TREE does not fail, 1. has been proved. Otherwise $E \equiv \bot$, and therefore $\overline{X} \approx \overline{t} \wedge \neg\exists\overline{V}E \equiv_{\mathcal{RH}} \overline{X} \approx \overline{t}$, proving 2. ■

Lemma 14 *Let $C \equiv \Pi_1\,\exists\overline{X}\,\Pi_2\,\exists\overline{\nu}\,\Pi_3(\overline{X} \approx \overline{t}(\overline{\nu}) \wedge C')$, where the variables in $\overline{X}$ are all distinct, no arithmetical operator occurs in $\overline{t}$ and no variable in $\overline{X}$ occurs in C' nor in $\overline{t}$. Then, $C \equiv_{\mathcal{RH}} \Pi_1\,\exists\overline{\nu}\,\Pi_2\Pi_3(C')$.*

Intuitively this lemma holds because, once the variables $\overline{X}$ are interpreted by specific values, the values interpreting the variables $\overline{\nu}$ are already fixed.

Lemma 15 *Let $C \equiv \exists\overline{X}\Pi(\neg(b_1 \wedge p_1) \wedge \ldots \wedge \neg(b_n \wedge p_n) \wedge p)$, where the set of eliminable variables of each basic formula b_i is a subset of $\overline{X}$ and Π binds no symbolic variable. Let $\{j_1, \ldots, j_s\} \subseteq \{1, \ldots, n\}$ be the set of indexes of those b_i with the form $\exists\overline{U}(\overline{Y} \approx \overline{U})$, where $\overline{U}$ contains no repetition of variables. Then, $C \equiv_{\mathcal{RH}} \Pi(\neg p_{j_1} \wedge \ldots \wedge \neg p_{j_s} \wedge p)$.*

Proof: Let $C' \equiv \Pi(\neg p_{j_1} \wedge \ldots \wedge \neg p_{j_s} \wedge p)$. Our goal is to prove both *a)* $C \vdash_{\mathcal{RH}} C'$ and *b)* $C' \vdash_{\mathcal{RH}} C$.

a) $C \vdash_{\mathcal{RH}} \Pi(\neg(b_{j_1} \wedge p_{j_1}) \wedge \ldots \wedge \neg(b_{j_s} \wedge p_{j_s}) \wedge p)$ is clear, since the formula in the right hand side is just the result of removing some members of the conjunction in C. But $b_{j_l} \equiv_{\mathcal{RH}} \top$ for each $1 \leq l \leq s$, therefore $C \vdash_{\mathcal{RH}} C'$.

b) Let $\{k_1,\ldots,k_{n-s}\} = \{1,\ldots,n\}\backslash\{j_1,\ldots,j_s\}$, and let b_i have the form $\exists\overline{U}^i(\overline{X}^i \approx \overline{t}^i)$. The way in which $\{j_1,\ldots,j_s\}$ has been defined, together with the fact that a denumerable set of symbolic constants is available, guarantees that there exists a $\overline{t}$ with the following property. For every $1 \leq i \leq n-s$, $(\overline{X}^i \approx \overline{t}^i \wedge \overline{X} \approx \overline{t}) \equiv_{\mathcal{RH}} \bot$. That is to say, intuitively, that $\overline{t}$ and $\overline{t}^i$ cannot be unified. A possible choice for $\overline{t}$ is a vector of distinct symbolic constants not occurring in C. Then

$$C' \vdash_{\mathcal{RH}} \Pi(\neg p_{j_1} \wedge \ldots \wedge \neg p_{j_s} \wedge p \wedge \neg\exists\overline{U}^{k_1}(\overline{X}^{k_1} \approx \overline{t}^{k_1} \wedge \overline{X} \approx \overline{t} \wedge p_{k_1}) \wedge \ldots$$

$$\wedge\neg\exists\overline{U}^{k_{n-s}}(\overline{X}^{k_{n-s}} \approx \overline{t}^{k_{n-s}} \wedge \overline{X} \approx \overline{t} \wedge p_{k_{n-s}})).$$

Such $\overline{t}$ plays the role of a witness, and using the properties of $\vdash_{\mathcal{RH}}$ it follows that

$$C' \vdash_{\mathcal{RH}} \exists\overline{X}\Pi(\neg p_{j_1} \wedge \ldots \wedge \neg p_{j_s} \wedge p \wedge \neg(b_{k_1} \wedge p_{k_1}) \wedge \ldots \wedge \neg(b_{k_{n-s}} \wedge p_{k_{n-s}})),$$

which is exactly what needed to be proved. ■

Lemma 16 *Let $C \equiv \exists\overline{X}\Pi_r((\exists\overline{U}^1(\overline{X} \approx \overline{t}^1) \wedge E_1) \vee \ldots \vee (\exists\overline{U}^m(\overline{X} \approx \overline{t}^m) \wedge E_m))$ be the completion for an elemental constraint, where Π_r binds every real variable in the body of C. Let*

$$IS = \{\emptyset \neq S \subseteq \{1,\ldots,m\} |\ \text{there is a symbolic unifier for } \{\overline{t}^i \approx \overline{t}^j\}_{i,j\in S}\},$$

and for each $S \in IS$ let $C_S \equiv \bigvee_{j\in S}(\overline{X} \approx \overline{t}^j \wedge E_j)$.

Then, $C \equiv_{\mathcal{RH}} \bigvee_{S\in IS} \exists\overline{X}\exists\overline{w_S}\Pi_r\exists\overline{U}(C_S\mu_S)$, where $\overline{U} = \overline{U}_1\ldots\overline{U}_m$, and μ_S is any most general n.a.o–unifier for $\{\overline{t}^i \approx \overline{t}^j\}_{i,j\in S}$ such that no variable of the set $\overline{w_S}$ of real variables in the range of μ_S occurs in C.

Proof: (sketch) Let $C' \equiv \bigvee_{S\in IS} \exists\overline{X}\exists\overline{w_S}\Pi_r\exists\overline{U}(C_S\mu_S)$. As C and C' are both sentences, in order to prove $C \equiv_{\mathcal{RH}} C'$ it is enough to prove that $\vdash_{\mathcal{RH}} C$ iff $\vdash_{\mathcal{RH}} C'$. Let us show $\vdash_{\mathcal{RH}} C$ implies $\vdash_{\mathcal{RH}} C'$ (the other implication is easier).

$\vdash_{\mathcal{RH}} C$ implies that there exists a ground symbolic substitution μ for $\overline{X}$ such that $\vdash_{\mathcal{RH}} \exists\overline{z}\Pi_r((\exists\overline{U}^1(\overline{X}\mu \approx \overline{t}^1) \wedge E_1\mu) \vee \ldots \vee (\exists\overline{U}^m(\overline{X}\mu \approx \overline{t}^m) \wedge E_m\mu))$, where $\overline{z}$ is the set of variables in the range of μ. Let S be the set of indexes i such that $\overline{X}\mu \approx \overline{t}^i$ is not $\mathcal{RH}$-equivalent to $\bot$. I.e., SOLVE-TREE does not fail if it is applied to $\overline{X}\mu \approx \overline{t}^i$. It is clear that $\vdash_{\mathcal{RH}} \exists\overline{z}\Pi_r(\bigvee_{i\in S}(\exists\overline{U}^i(\overline{X}\mu \approx \overline{t}^i) \wedge E_i\mu))$. Let $\overline{U} \equiv U_1\ldots U_m$. Then $\vdash_{\mathcal{RH}} \exists\overline{z}\Pi_r\exists\overline{U}(\bigvee_{i\in S}(\overline{X}\mu \approx \overline{t}^i) \wedge E_i\mu)$. Let μ_S be any most general n.a.o–unifier for $\{\overline{t}^i \approx \overline{t}^j\}_{i,j\in S}$. Then, $\vdash_{\mathcal{RH}} \exists\overline{z}\exists\overline{w_S}\Pi_r\exists\overline{U}(\bigvee_{i\in S}(\overline{X}\mu \approx \overline{t}^i\mu_S) \wedge E_i\mu)$. So finally

$$\vdash_{\mathcal{RH}} \exists\overline{X}\exists\overline{w_S}\Pi_r\exists\overline{U}(\bigvee_{i\in S}(\overline{X} \approx \overline{t}^i\mu_S) \wedge E_i) \equiv_{\mathcal{RH}} \exists\overline{X}\exists\overline{w_S}\Pi_r\exists\overline{U}(C_S\mu_S).$$

Thus $\vdash_{\mathcal{RH}} \bigvee_{S\in IS} \exists\overline{X}\exists\overline{w_S}\Pi_r\exists\overline{U}(C_S\mu_S) \equiv C'$. ■

5.2 The Procedure *Elemental-Sat*

For the sake of a better understanding, before dealing with the algorithm that is able to decide the $\mathcal{RH}$-satisfiability of any elemental constraint $\Pi(D_1 \vee \ldots \vee D_m)$, we present a more simple procedure that may be used when $m = 1$.

ELEMENTAL-SAT (case $m = 1$)
`Input:` An elemental constraint $C \equiv \Pi D_1$.
`Output`: $\top$ if C is $\mathcal{RH}$-satisfiable. $\bot$ otherwise.
`Procedure`: The sequential application of the following steps should be carried out.

1) Replace C by its completion.

   Now C has the form ΠD_1, where $D_1 \equiv b \wedge \neg b_1 \wedge \ldots \wedge \neg b_n \wedge p$. Let Π_r be the sequence of real quantifications in Π, and $\overline{y}$ be the set of variables bound by Π_r. Let $\overline{X}$ be the eliminable variables of D_1, and $\overline{x}$ the set of free real variables in C. Let $C_1 \equiv \exists \overline{X} \exists \overline{x} \Pi_r D_1$ be the existential closure of C.

2) If no symbolic variable occurs in C, then C' is already a Tarski sentence, so go directly to step 5).

   Otherwise, let $b \equiv \exists \overline{U}(\overline{X} \approx \overline{t}(\overline{U}, \overline{x}, \overline{y}))$. Replace in $\overline{t}(\overline{U}, \overline{x}, \overline{y})$ each real subterm $s(\overline{x}, \overline{y})$ by a new real variable z, obtaining $\overline{t}'(\overline{U}, \overline{z})$.

   Build $C_2 \equiv \exists \overline{X} \exists \overline{x} \Pi_r \exists \overline{z} \exists \overline{U}(\overline{X} \approx \overline{t}'(\overline{U}, \overline{z}) \wedge \neg b_1 \wedge \ldots \wedge \neg b_n \wedge p \wedge \overline{z} \approx \overline{s}(\overline{x}, \overline{y}))$.

3) For each $1 \leq i \leq n$ proceed as follows. Write $b_i \equiv \exists \overline{U}^i(\overline{X} \approx \overline{t_i})$, and apply SOLVE-TREE to $\overline{t_i} \approx \overline{t}'(\overline{U}, \overline{z})$, obtaining the output $E \wedge p'_i$, and let $b'_i \equiv \exists \overline{U}^i E$. Remove in b'_i the equalities with left hand side in $\overline{U}^i$, together with the existential quantifiers over them.

   Build $C_3 \equiv \exists \overline{U} \exists \overline{z} \exists \overline{x} \Pi_r(\neg(b'_1 \wedge p'_1) \wedge \ldots \wedge \neg(b'_n \wedge p'_n) \wedge p \wedge \overline{z} \approx \overline{s}(\overline{x}, \overline{y}))$.

   If for j SOLVE-TREE failed, then $\neg(b'_j \wedge p'_j)$ is not included in C_3.

4) Let $\{j_1, \ldots, j_s\} \subseteq \{1, \ldots, n\}$ be the set of indexes of those b_i with the form $\exists \overline{U}(\overline{Y} \approx \overline{U})$ occurring in C_3, where $\overline{U}$ contains no variable repetition.

   Build $C_4 \equiv \exists \overline{z} \exists \overline{x} \Pi_r(\neg p'_{j_1} \wedge \ldots \wedge \neg p'_{j_s} \wedge p \wedge \overline{z} \approx \overline{s}(\overline{x}, \overline{y}))$.

5) Use a *CAD*-based procedure to eliminate the quantifiers of C_4, producing $C_5 \in \{\top, \bot\}$.

Proposition 17 (Total Correctness) *For a given input elemental constraint C, the algorithm* ELEMENTAL-SAT *(case $m = 1$) terminates and the output produced is $\top$ if C is $\mathcal{RH}$-satisfiable, and $\bot$ otherwise.*

Proof: C is $\mathcal{RH}$-equivalent to its completion thanks to Lemma 12, and thus it is $\mathcal{RH}$-satisfiable if and only if $C_1 \equiv_{\mathcal{RH}} \top$. Therefore in order to prove the claim it is enough to show that the formulas $C_i, 1 \leq i \leq 5$, are pairwise $\mathcal{RH}$-equivalent.

- $C_1 \equiv_{\mathcal{RH}} C_2$, due to the properties of $\vdash_{\mathcal{RH}}$.

- By virtue of Lemma 13, $C_2 \equiv_{\mathcal{RH}}$

$$\exists\overline{X}\exists\overline{x}\Pi_r\exists\overline{z}\exists\overline{U}(\overline{X} \approx \overline{t}'(\overline{U},\overline{z}) \wedge \neg(b'_1 \wedge p'_1) \wedge \ldots \wedge \neg(b'_n \wedge p'_n) \wedge p \wedge \overline{z} \approx \overline{s}(\overline{x},\overline{y})).$$

  It must be noticed that, for $1 \leq i \leq n$, the constraint $b'_i \wedge p'_i$ does not contain any occurrences of $\overline{X}$. Hence the conditions in Lemma 14 hold, and so the latter constraint is $\mathcal{RH}$-equivalent to C_3.

- By virtue of Lemma 15, $C_3 \equiv_{\mathcal{RH}} C_4$.

- The *CAD*-based quantifier-elimination methods preserve $\mathcal{RH}$-equivalence, so $C_4 \equiv_{\mathcal{RH}} C_5$. ■

At this stage we are ready to present the general procedure ELEMENTAL-SAT. Its first steps are analogous to those in the case $m = 1$, and many of the ideas applied in it are still useful.

ELEMENTAL-SAT
`Input`: An elemental constraint $C \equiv \Pi(D_1 \vee \ldots \vee D_m)$.
`Output`: $\top$ if C is $\mathcal{RH}$-satisfiable. $\bot$ otherwise.
`Procedure`: The sequential application of the following steps should be carried out.

1) Replace C by its completion. C has the form $\Pi(D_1 \vee \ldots \vee D_m)$, where $D_j \equiv b^j \wedge \neg b^j_1 \wedge \ldots \wedge \neg b^j_{n_j} \wedge p^j$ for every j, $1 \leq j \leq m$. Let Π_r be the sequence of real quantifications in Π, and $\overline{y}$ be the set of variables bound by Π_r. Let $\overline{X}$ be the eliminable variables of D_1, and $\overline{x}$ the set of free real variables in C. Let $C_1 \equiv \exists\overline{X}\exists\overline{x}\Pi_r(D_1 \vee \ldots \vee D_m)$ be the existential closure of C.

2) For each $1 \leq j \leq m$, proceed with the following steps:

   2.1) Let $b^j \equiv \exists\overline{U}^j(\overline{X} \approx \overline{t}^j(\overline{U}^j,\overline{x},\overline{y}))$. Replace in $\overline{t}^j(\overline{U}^j,\overline{x},\overline{y})$ each real subterm $s^j(\overline{x},\overline{y})$ by a new real variable z^j, obtaining $\overline{t}^{j\prime}(\overline{U}^j,\overline{z})$.
   Build the constraint

$$C^j_{2.1} \equiv \exists\overline{z}^j(\overline{X} \approx \overline{t}^{j\prime}(\overline{U}^j,\overline{z}^j) \wedge \neg b^j_1 \wedge \ldots \wedge \neg b^j_n \wedge p^j \wedge \overline{z}^j \approx \overline{s}^j(\overline{x},\overline{y})).$$

   2.2) For $1 \leq i \leq n_j$, write $b^j_i \equiv \exists\overline{U}^j_i(\overline{X} \approx \overline{t}^j_i)$, and apply SOLVE-TREE to $\overline{t}^j_i \approx \overline{t}^{j\prime}(\overline{U}^j,\overline{z}^j)$, producing $E_i \wedge p^{j\prime}_i$. Let $b^{j\prime}_i \equiv \exists\overline{U}^j_i E_i$.
   Remove in $b^{j\prime}_i$ the equalities with left hand side in $\overline{U}^j_i$, together with the existential quantifiers over them. Build the constraint

$$C^j_{2.2} \equiv \exists\overline{z}^j(\overline{X} \approx \overline{t}^{j\prime}(\overline{U}^j,\overline{z}^j) \wedge \neg(b^{j\prime}_1 \wedge p^{j\prime}_1) \wedge \ldots \wedge \neg(b^{j\prime}_{n_j} \wedge p^{j\prime}_{n_j}) \wedge p^j \wedge \overline{z^j} \approx \overline{s}^j(\overline{x},\overline{y})).$$

3) Compute

$$IS = \{\emptyset \neq S \subseteq \{1,\ldots,m\} \mid \text{there is a symbolic unifier for } \{\overline{t}^{i\prime} \approx \overline{t}^{j\prime}\}_{i,j\in S}\}.$$

4) For every $S \in IS$, set $check_S = \bot$. Let $C_4 = \bot$.

5) While $C_4 = \bot$ and there is $S \in IS$ such that $check_S = \bot$ proceed with the following steps.

5.1) Select $S \in IS$ such that $check_S = \bot$.

For the sake of simplicity, let us assume that $S = \{1, \ldots, l\}$, where $l \leq m$. Calculate a most general n.a.o–unifier μ_S associated to S, and let $\overline{w}$ be the set of real variables occurring in its range. For every $i, 1 \leq i \leq l$, $\overline{t}^{i\prime}(\overline{U}^i, \overline{z}^i)\mu \equiv_h \overline{t}^{1\prime}(\overline{U}^1, \overline{z}^1)\mu$. Build the polynomial p^i_{unif} that consists on the conjunction of all polynomials $s_1 \approx s_2$, where $s_1 : r$ is a subterm of $\overline{t}^{i\prime}(\overline{U}^i, \overline{z}^i)\mu$ and $s_2 : r$ is the subterm of $\overline{t}^{1\prime}(\overline{U}^1, \overline{z}^1)\mu$ in the same position.

Build the constraint

$$C^S_{5.1} \equiv \exists\overline{X}\exists\overline{w}\exists\overline{x}\Pi_r\exists\overline{U}\exists\overline{z}(\overline{X} \approx \overline{t}^{1\prime}(\overline{U}^1, \overline{z}^1)\mu_S \wedge$$

$$(\neg(b_1^{1\prime} \wedge p_1^{1\prime}) \wedge \ldots \wedge \neg(b_{n_1}^{1\prime} \wedge p_{n_1}^{1\prime}) \wedge p^1 \wedge \overline{z}^1 \approx \overline{s}^1(\overline{x}, \overline{y}) \wedge p^1_{unif}) \vee \ldots$$

$$\vee(\neg(b_1^{l\prime} \wedge p_1^{l\prime}) \wedge \ldots \wedge \neg(b_{n_l}^{l\prime} \wedge p_{n_l}^{l\prime}) \wedge p^l \wedge \overline{z}^l \approx \overline{s}^l(\overline{x}, \overline{y}) \wedge p^l_{unif})),$$

where $\overline{U} = \overline{U}^1 \ldots \overline{U}^m$ and $\overline{z} = \overline{z}^1 \ldots \overline{z}^m$.

5.2) Move the quantifiers $\exists\overline{U}$ and $\exists\overline{z}$ and place it next to $\exists\overline{X}$. Eliminate $\exists\overline{X}$ and remove the equalities in which such variables occur. In each $b^{k\prime}_j$ replace $\overline{X}$ by $\overline{t}^{1\prime}(\overline{U}^1, \overline{z}^1)\mu_S$. Use SOLVE-TREE with each $b^{k\prime}_j \wedge p^{k\prime}_j$ (excluding the quantifiers prefix), and let $\neg(b^{k\prime\prime}_j \wedge p^{k\prime\prime}_j)$ be the constraint produced. Thus, the following constraint must be built:

$$C^S_{5.2} \equiv \exists\overline{U}\exists\overline{z}\exists\overline{w}\exists\overline{x}\Pi_r($$

$$(\neg(b_1^{1\prime\prime} \wedge p_1^{1\prime\prime}) \wedge \ldots \wedge \neg(b_{n_1}^{1\prime\prime} \wedge p_{n_1}^{1\prime\prime}) \wedge p^1 \wedge \overline{z}^1 \approx \overline{s}^1(\overline{x}, \overline{y}) \wedge p^1_{unif}) \vee \ldots$$

$$\vee(\neg(b_1^{l\prime\prime} \wedge p_1^{l\prime\prime}) \wedge \ldots \wedge \neg(b_{n_l}^{l\prime\prime} \wedge p_{n_l}^{l\prime\prime}) \wedge p^l \wedge \overline{z}^l \approx \overline{s}^l(\overline{x}, \overline{y}) \wedge p^l_{unif})).$$

If for j, k SOLVE-TREE failed, then $\neg(b^{k\prime\prime}_j \wedge p^{k\prime\prime}_j)$ must not be included in $C_{5.2}$.

5.3) Proceed as in step 4) of the case $m = 1$, removing every basic formula and symbolic quantifier, leaving only those polynomials associated to basic formulas whose right hand sides are different variables. Let $C_{5.3}$ be the result:

$$C^S_{5.3} \equiv \exists\overline{z}\exists\overline{w}\exists\overline{x}\Pi_r((\neg p^1_{j^1_1}{}'' \wedge \ldots \wedge \neg p^1_{j^1_{n_1}}{}'' \wedge p^1 \wedge \overline{z}^1 \approx \overline{s}^1(\overline{x}, \overline{y}) \wedge p^1_{unif}) \vee \ldots$$

$$\vee(\neg p^l_{j^l_1}{}'' \wedge \ldots \wedge \neg p^l_{j^l_{n_l}}{}'' \wedge p^l \wedge \overline{z}^l \approx \overline{s}^l(\overline{x}, \overline{y}) \wedge p^l_{unif})).$$

5.4) Apply a *CAD*-based method to eliminate the quantifiers in $C_{5.3}$, producing $\top$ or $\bot$. Store the result in C_4, and set $check_S = \top$.

6) Return C_4.

Proposition 18 (Total Correctness) *For a given input elemental constraint C, the algorithm* ELEMENTAL-SAT *terminates and the output produced is $\top$ if C is $\mathcal{RH}$-satisfiable, and $\bot$ otherwise.*

Proof: C is $\mathcal{RH}$-equivalent to its completion thanks to Lemma 12, and thus it is $\mathcal{RH}$-satisfiable if and only if $C_1 \equiv_{\mathcal{RH}} \top$. Analogously as it was argued in the proof for the case $m = 1$, $D_j \equiv_{\mathcal{RH}} C^j_{2.1} \equiv_{\mathcal{RH}} C^j_{2.2}$, $1 \leq j \leq m$, thanks to Lemma 13. Therefore, $C_1 \equiv_{\mathcal{RH}} \exists\overline{X}\exists\overline{x}\Pi_r(\exists\overline{U}^1 C^1_{2.2} \vee \ldots \vee \exists\overline{U}^m C^m_{2.2})$. By virtue of Lemma 16, that is $\mathcal{RH}$-equivalent to

$$\bigvee_{S \in IS} \exists\overline{X}\exists\overline{x}\Pi_r\exists\overline{U}(C_S\mu_S), \qquad (\dagger)$$

where $C_S \equiv \bigvee_{j \in S} C^j_{2.2}$ for each $S \in IS$ and $\overline{U} = \overline{U}_1 \ldots \overline{U}_m$. Let us prove now that $(\dagger) \equiv_{\mathcal{RH}} \bigvee_{S \in IS} C^S_{5.4} \equiv_{\mathcal{RH}} C_4$. We know that $(\dagger) \equiv_{\mathcal{RH}} \top$ if and only if there is $S \in IS$ such that $\exists\overline{X}\exists\overline{x}\Pi_r\exists\overline{U}(C_S\mu_S) \equiv_{\mathcal{RH}} \top$. It must be noticed that the loop in step 5) is defined in such a way that $C_4 \equiv_{\mathcal{RH}} \top$ if and only if there is $S \in IS$ such that $C^S_{5.3} \equiv_{\mathcal{RH}} \top$. The latter holds because the *CAD*-method used preserves $\mathcal{RH}$-equivalence. Therefore, it is enough to prove that $\exists\overline{X}\exists\overline{x}\Pi_r\exists\overline{U}(C_S\mu_S) \equiv_{\mathcal{RH}} C^S_{5.3}$ for every $S \in IS$, which can be shown as follows:

- For any fixed S, $\exists\overline{X}\exists\overline{x}\Pi_r\exists\overline{U}(C_S\mu_S) \equiv_{\mathcal{RH}} C^S_{5.1}$, mainly because $p^i_{unif} \vdash_{\mathcal{RH}} \overline{t}^{i'}(\overline{U}^i, \overline{z}^i)\mu_S \approx \overline{t}^{1'}(\overline{U}^1, \overline{z}^1)\mu_S$.
- Analogously as in the proof for the case $m = 1$, $C^S_{5.1} \equiv_{\mathcal{RH}} C^S_{5.2}$ by virtue of Lemmas 13 and 14.
- The constraint $C^S_{5.3}$ is obtained from $C^S_{5.2}$ in an analogous way as in the step 4) of the proof for the case $m = 1$. Again, by virtue of Lemma 15, such transformation preserves $\mathcal{RH}$-equivalence. ■

The following example illustrates the usage of ELEMENTAL-SAT with a non-trivial constraint.

Example 5 Suppose that we need to decide whether the elemental constraint C below is $\mathcal{RH}$-satisfiable.

$$C \equiv \exists X\exists Y\exists y\forall x(D_1 \vee D_2 \vee D_3),$$

where

$D_1 \equiv \exists U_1 V_1(X \approx f(0) \wedge Y \approx g(U_1, V_1)) \wedge \neg\exists W_1(X \approx f(x) \wedge Y \approx W_1)$,
$D_2 \equiv \exists U_2 V_2(X \approx U_2 \wedge Y \approx g(f(1), V_2)) \wedge \neg\exists W_2(X \approx f(0) \wedge Y \approx g(f(x), W_2))$,
$D_3 \equiv \exists U_3 V_3(X \approx U_3 \wedge Y \approx V_3) \wedge x(x-1) \approx y$.

So, in this example $m = 3$. Steps 1) is not needed this time, since the input is already the completion of a quantified $\mathcal{RH}$-normal form. Step 2) replaces D_1 by

$$\exists U_1 V_1(X \approx f(z_1) \wedge Y \approx g(U_1, V_1)) \wedge \neg\exists W_1(X \approx f(x) \wedge Y \approx W_1) \wedge z_1 \approx 0$$

and D_2 by

$$\exists U_2 V_2(X \approx U_2 \wedge Y \approx g(f(z_2), V_2)) \wedge \neg\exists W_2(X \approx f(0) \wedge Y \approx g(f(x), W_2)) \wedge z_2 \approx 1.$$

D_3 does not have any real constants or operators in its positive basic formula, so step 2) is not applied to it.

In step 3) IS is found to be

$$IS = \{\{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}.$$

Suppose that the first S selected in step 5.1) is $S = \{1\}$. Then the procedure works as follows.

Initial constraint, $\overline{\exists}(C_s\mu_s)$:
$\exists X \exists Y \exists z_1 \exists y \forall x (\exists U_1 V_1 (X \approx f(z_1) \wedge Y \approx g(U_1, V_1)) \wedge \neg \exists W_1 (X \approx f(x) \wedge Y \approx W_1) \wedge z_1 \approx 0)$.

5.1) Since there is only one formula in the disjunction, no change is performed.

5.2) Reordering quantifiers, removing X and Y, and applying SOLVE-TREE:
$\exists z_1 U_1 V_1 y \forall x (\neg (\exists U_1' V_1' (U_1 \approx U_1' \wedge V_1 \approx V_1') \wedge z_1 \approx x) \wedge z_1 \approx 0)$.

5.3) Removing basic formulas $\mathcal{RH}$-equivalent to $\top$:
$\exists z_1 y \forall x (\neg z_1 \approx x \wedge z_1 \approx 0)$.

5.4) Using a *CAD*-based quantifier-elimination method with the previous Tarski sentence, it turns out to be $\mathcal{RH}$-equivalent to $\bot$.

Therefore, C_4 is still false, and another $S \in IS$ must be tried. For example, let us assume that $S = \{1, 2\}$ is selected.

Initial constraint $\overline{\exists}(C_s\mu_s)$:

$\exists X \exists Y \exists y \exists x \forall x [$
$(\exists U_1 V_1 (X \approx f(z_1) \wedge Y \approx g(U_1, V_1)) \wedge \neg \exists W_1 (X \approx f(x) \wedge Y \approx W_1) \wedge z_1 \approx 0) \vee$
$(\exists U_2 V_2 (X \approx U_2 \wedge Y \approx g(f(z_2), V_2)) \wedge \neg \exists W_2 (X \approx f(0) \wedge Y \approx g(f(x), W_2)) \wedge z_2 \approx 1)]$.

5.1) Applying a most general symbolic unifier for the right hand sides of the equalities for X, Y.
$\exists X Y V_1 z_1 z_2 y \forall x [z_1 \approx 0 \wedge z_2 \approx 1 \wedge (X \approx f(z_1) \wedge Y \approx g(f(z_2), V_1)) \wedge$
$(\neg \exists W_1 (X \approx f(x) \wedge Y \approx W_1) \vee \neg \exists W_2 (X \approx f(0) \wedge Y \approx g(f(x), W_2)))]$.

5.2) Elimination of X, Y. Application of SOLVE-TREE:
$\exists V_1 z_1 z_2 y \forall x [z_1 \approx 0 \wedge z_2 \approx 1 \wedge (\neg (\exists V_1' (V_1 \approx V_1') \wedge x \approx z_1) \vee$
$\neg (\exists V_1'' (V_1 \approx V_1'') \wedge z_1 \approx 0 \wedge z_2 \approx x))]$.

5.3) Removing basic formulas $\mathcal{RH}$-equivalent to $\top$:
$\exists z_1 z_2 y \forall x [z_1 \approx 0 \wedge z_2 \approx 1 \wedge (\neg x \approx z_1 \vee \neg (z_1 \approx 0 \wedge z_2 \approx x))]$.

5.4) Using a *CAD*-based quantifier-elimination method with the previous Tarski sentence, it turns out to be $\mathcal{RH}$-equivalent to $\top$. That is in fact easy to see, since a straightforward simplification renders $\forall x (\neg x \approx 0 \vee \neg x \approx 1)$.

Therefore, C_4 is set to $\top$, and that is the output of ELEMENTAL-SAT.

6 Conclusion

The constraint system $\mathcal{RH}$ has been defined joining the axiomatization of the algebra of finite trees together with the axiomatization for real closed fields. Both theories are decidable, and our interest was to find a decision procedure for their combination. The framework of the constraint system $\mathcal{RH}$ is the *CLP* scheme, and it can be considered as a domain that produces a particular instance. In this field, there is a variety of works dealing with different constraint domains [11, 2, 5, 9]. Our contribution relies on the fact that we have dealt with a harder, more general satisfiability problem, because, having the domain in the context of a logic programming language based on hereditary Harrop formulas, any occurrence of existential and universal quantifiers is allowed in the constraints, instead of only existential ones as in Horn clauses. On the other hand, comparing our method with the decision procedure for combined theories proposed in [18], it is important to realize that the latter applies only to quantifier-free formulas, and therefore the technique of propagation of equalities on which it relies does not seem useful for the purpose of the present paper. This is due to the fact that such propagation incorporates equalities implied by polynomials, but does not replace them, which does not help to the elimination of quantifiers. However, it may be used to check $\mathcal{RH}$-satisfiability of constraints with no mixed quantifier prefix, although the solving procedure described in this paper is also able to do it.

Choosing as starting point the decision procedure due to Maher [15] for the theories of the algebras of finite, rational and infinite trees, based on elimination of quantifiers, we have extended it to dealing with quantifiers over real variables, using *CAD* based techniques [19, 3]. The incorporation of real variables and polynomials to the formulas to be treated is not trivial at all, and its effect on the original algorithm due to Maher has been studied. A procedure to solve a subclass of the set of $\mathcal{RH}$-constraints has been defined based on the reduction of the original constraints to simpler ones, for which a satisfiability decision method has also been described. Such procedure can be regarded as the basis of a solver for the constraint logic programming language *HH($\mathcal{RH}$)*. In this paper we have focused on its foundations, which are proved in an original way. We tried to provide procedures as simple and natural as possible, in order to make it more understandable and easy to study. However, for a particular implementation, many refinements and improvements, including the incorporation of heuristics, would be most convenient. With respect to the efficiency, the algorithm is very naïve, since it incorporates parts like the SOLVE-TREE algorithm, whose complexity is exponential. The tasks concerning elimination of real quantifiers computing a *CAD* have polynomial complexity [3], but some calculations may be reused when the next real quantifier is being eliminated. Another desirable feature concerning the implementation of the solver in the context of *HH($\mathcal{RH}$)* is the incrementality. The goal-solving procedure presented in [13] handles prenex constraints as partial calculated answers, which evidently benefits the phase of preprocessing, although it must be studied how that may be useful for the quantifier elimination phase, in order to profit the calculus performed for the constraints of previous steps.

Our interest as future related research consists in the dealing, if possible, with a greater class of $\mathcal{RH}$-constraints (including such of Example 3), as well as to formally analyze the complexity of the algorithm SIMPLIFY. We are working in the implementation of a $\mathcal{RH}$-constraint solver; with that propose, we are looking for techniques that could improve the efficiency of our theoretical solver, and for particular implementations dealing with certain classes of polynomials.

Acknowledgements: We are grateful to Jesús Escribano for his collaboration on the part of this work referring real numbers.

References

[1] Buchberger, B. and , Winkler, F. (eds.) *Gröebner Bases and Applications*, Cambridge Univ. Press, 1998.

[2] Caprotti, O. *Extending RISC-CLP(Real) to Handle Symbolic Functions*, A. Miola (ed.) DISCO 1993. LNCS 722, Springer, 1993, 241–255.

[3] Caviness, B.F. and Johnson, J.R. *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Springer, 1998.

[4] Clark, K.L., Negation as Failure, in: H. Gallaire and J. Minker (eds.) *Logic and Databases* 293-322, Plenum Press, 1978.

[5] Colmerauer, A. *An introduction to PROLOG III*, Commun. ACM 33(7), 1990, 69–90.

[6] Comon, H. and Lescanne, P. *Equational problems and disunification*, J. of Symbolic Computation 7, 1989, 371–425.

[7] García-Díaz, M. and Nieva, S. *Solving mixed quantified constraints over a domain based on real numbers and Herbrand terms*, Hu, Z. and Rodríguez-Artalejo, M. (eds.) Functional and Logic Programming, FLOPS'02. LNCS 2441, Springer, 2002, 103-118.

[8] Herbrand, J. *Researches sur la theorie de la demonstration.* In: Ecrits logiques de Jacques Herbrand, Paris, PUF, 1968.

[9] Hong, M., *RISC-CLP(CF) Constraint Logic Programming over Complex Functions*, Frank Pfenning (ed.) Logic Programming and Automated Reasoning, LPAR'94. LNCS 822, Springer, 1994, 99-113.

[10] Jaffar, J. and Maher, M. *Constraint Logic Programming: A Survey*, J. of Logic Programming 19(20), 1994, 503–581.

[11] Jaffar, J., Michaylov, S., Stuckey, P. and Yap, R. *The CLP(R) Language and System* , ACM Transactions on Programming Languages 14(3), 1992, 339–395.

[12] Leach, J. and Nieva, S. *A Higher-Order Logic Programming Language with Constraints*, Kuchen, H. and Ueda, K. (eds.) FLOPS'01. LNCS 2024, Springer, 2001, 102–122.

[13] Leach, J., Nieva, S. and Rodríguez-Artalejo, M. *Constraint Logic Programming with Hereditary Harrop Formulas*, Theory and Practice of Logic Programming 1(4), Cambridge University Press, 2001, 409–445.

[14] Lloyd, J. W. *Foundations of Logic Programming*, Springer-Verlag, 1987.

[15] Maher, M. *Complete axiomatizations of the algebras of finite, rational and infinite trees*, in Procs. of the Third Annual Symposium on Logic in Computer Science, Edinburgh, 1988. IEEE Computer Society, 348–357.

[16] Miller, D., Nadathur, G., Pfenning, F. and Scedrov, A. *Uniform Proofs as a Foundation for Logic Programming*, Annals of Pure and Applied Logic 51, 1991, 125–157.

[17] Nadathur, G. *A Proof Procedure for the Logic of Hereditary Harrop Formulas*, J. of Automated Reasoning 11, 1993, 111–145.

[18] Nelson, G. and Oppen, D. *Simplification by Cooperating Decision Procedures*, ACM Transactions on Programming Languages and Systems 1(2), 1979, 245–257.

[19] Tarski, A. *A Decision Method for Elementary Algebra and Geometry*, University of California Press, 1951.