Solving Mixed Quantified Constraints 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. The implementation of this new language requires the ability to test the satisfiability of constraints built up by means of terms and predicates belonging to the domain of the chosen constraint system, and by the connectives and quantifiers usual in first-order logic. In this paper we present a constraint system called $\mathcal{RH}$ for a hybrid domain that mixes Herbrand terms and real numbers. It arises when joining the axiomatization of the arithmetic of real numbers and the axiomatization of the algebra of finite trees. We have defined an algorithm to solve certain constraints of this kind. The novelty relies on the combination of two different mechanisms, based on elimination of quantifiers, one used for solving unification and disunification problems, the other used to solve polynomials. This combination provides 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 (based on Horn clauses) 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 the set of valid answer constraints for any goal, which will depend on the specific domain of the constraint system.

This kind of extension is also dealt with in the papers [13] and [12] but, in both cases, the base language is the logic of Hereditary Harrop formulas *HH* (first-order and higher-order, respectively). This 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 using unification, is no longer valid in 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*. But, in *HH(C)*, the extension of *HH* including constraints, the technique to obtain answer substitutions using unification is replaced by the checking 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 all the connectives in first-order logic, equations between Herbrand terms and, moreover, other relations belonging to the constraint signature between terms of the instance domain.

Following one of the open research lines mentioned in [13], we intended to find decidable theories which could be the basis of a first-order constraint system for the abstract logic programming language *HH(C)*, but which are also useful programming domains. The 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. We must be able to solve equality constraints between such terms, but we also intend to deal with disequalities, and, of course, these relations may occur embedded in expressions together with several connectives and quantifiers, as reasoned before. This domain has been axiomatized and decision mechanisms have been defined for the corresponding theory [15, 6]. However, our approach demands an enlargement of it that provides more expressiveness, but simultaneously keeping the good properties and efficiency inherent to *CLP*, in which the tests for constraint satisfiability and the general goal solving procedure respect to a program are separated. The fields of application of *CLP* with constraint domains related to 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 the field of real numbers [19], this domain becomes a good choice as a constraint system instance. In this article we aimed to join the theory of the arithmetic of real numbers and the one of Herbrand terms, to produce a mixed constraint system called $\mathcal{RH}$ and so the instance *HH($\mathcal{RH}$)* of the language of *HH* with constraints. An algorithm to solve constraints for this language has been described. There are other works in the literature that introduce mixed numeric and symbolic domains (see e.g. [2, 5, 9]), but 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 article 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. The main results of the paper appear in Section 3. First we present two rules that constitute the core of a procedure to solve $\mathcal{RH}$-constraints. In order to give the foundations of these rules, several definitions and technical results are needed, their proofs appear in [7]. Then the algorithm that constitutes this procedure is detailed. In Section 4, its behavior is shown through examples. We end summarizing the contributions of our work, and stating future research, in Section 5.

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. In [13] a logic programming language called *HH(C)* is defined by adding constraints of a generic constraint system $\mathcal{C}$ to *HH*. In it, the language and the conditions of a system $\mathcal{C}$ to be a constraint system are specified, but no constraint solver for any particular system is studied. Here we will concentrate on an specific constraint system denoted by $\mathcal{RH} = \langle \mathcal{L}_{\mathcal{RH}}, \vdash_{\mathcal{RH}} \rangle$, 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 formula 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.
- Arithmetic operators $+, -, * : r \times r \rightarrow r$, and a finite or denumerable set of symbolic function $f : \tau_1 \times \ldots \times \tau_n \rightarrow h$, 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 relations of 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:

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

where $f : \tau_1 \times \ldots \times \tau_n \rightarrow h$ and $t_i : \tau_i, 1 \leq i \leq n$. Notice that infix notation is used for arithmetic 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 the goals forces the constraint system to be able to express and deduce complex constraints during the process of goal solving, that specifically contain a mixed prefix of quantifiers. So, we should impose to $\mathcal{L}_{\mathcal{RH}}$ to be closed under $\wedge, \vee, \exists, \forall$ and $\neg$. For instance, if the signature contains the symbols $cons : r \times h \rightarrow h$ and $nil : h$ for the list constructor and the empty list, respectively, the formula

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

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 any quantifier free constraint, such that its atomic subformulas are built up exclusively with real predicate symbols and real terms. 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 in t, or p, respectively. The substitution of a set of variables $\overline{\nu}$ by a set 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 $\overline{t}$ does not contain symbolic variables. It must be understood in the sequel that $\overline{X \approx t}$ stands for a (possibly empty) conjunction of equalities of the form $X \approx t$.

The deducibility relation between constraints, $\vdash_{\mathcal{RH}}$, is defined as the result of combining the classic deducibility relation with equality, $\vdash_{\approx}$, with the axiomatization for the field of real numbers due to Tarski [19], plus the axiomatization for the Herbrand universe due to Clark [4]. I. e., if $Ax_{\mathcal{R}}$ and $Ax_{\mathcal{H}}$ denote respectively the sets of axioms for these two theories, then, for 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 $\vdash_{\mathcal{RH}}$ verifies 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 when $\emptyset \vdash_{\mathcal{RH}} \overline{\exists} C$, where $\overline{\exists} C$ represents 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 $C' \vdash_{\mathcal{RH}} C$. $\vdash_{\mathcal{RH}}$ verifies 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 aim in this paper has been to find a procedure to determine whether a constraint of the system $\mathcal{RH}$ is $\mathcal{RH}$-satisfiable. It is important to notice that due to the mixed quantifier prefix binding real and symbolic variables, no separate solvers dealing with Herbrand terms and reals, respectively, can be used. Specifically, it is not 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.

3 A Solving Procedure for $\mathcal{RH}$-Constraints

The procedure that will be described is based 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 its $\mathcal{RH}$-satisfiability can be deduced. Some steps in this transformation make use exclusively of properties of the connectives in first-order logic; others are based on the axiomatization of the algebra of finite trees and of the field of real numbers; the essential part attempts to eliminate quantifiers. It combines the techniques of Maher [15] for symbolic quantified variables, with the elimination of real quantified variables using *C*ylindrical *A*lgebraic *D*ecomposition (*CAD*) [3].

We have found a condition, with respect to the order on the occurrence of the real and symbolic quantifications, which guarantees that the solving procedure finishes obtaining an elemental constraint. Nevertheless, the procedure can also happen to end in

some other cases for which that condition is not fulfilled. This is the case when the included polynomials can be simplified, by eliminating a variable that can be represented in function of the others by means of a finite number of equations. This problem can be decided using techniques based on Gröebner bases, see for instance the works included in [1].

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

3.1 Solved Forms, Basic Formulas and Elemental Constraints

In order to define elemental constraints, for which $\mathcal{RH}$-satisfiability may be decided, we previously introduce some concepts concerning constraints that are conjunctions of equalities and polynomials.

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

In the following, we describe an algorithm whose purpose is to find a solved form of a system of equalities and polynomial. It is based on the common rules for equality solving in the algebra of finite trees [8]. The algorithm, designated by *Solve-Tree*, simply consists in the sequential, nondeterministic application of the rules immediately presented, until none of them can be applied. These 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 the rule into the $\mathcal{RH}$-equivalent system E_2.

The Algorithm 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.

For any E used as input, the algorithm terminates, producing a solved form for E if there is any, and $\bot$ otherwise. In the latter case, we say that the algorithm *fails*. It must

also be noticed that 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, and that a system of equalities and polynomial 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.

Definition 3 *A* basic formula *is $\top$, or 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 set of boolean combinations of basic formulas *is defined as the minimum set closed under $\vee$, $\wedge$ and $\neg$ that contains every basic formula.*

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

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 $\Pi \Pi' C$, in which Π is a sequence of existential quantifications and Π' binds only real variables.*

We are able to decide the $\mathcal{RH}$-satisfiability of elemental constraints[1]. Therefore, the first aim of the $\mathcal{RH}$-constraint solver will be to reach elemental constraints 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. This normalization will be easily accomplished following the steps of the next algorithm.

$\mathcal{RH}$-Normalization

1. All negations are distributed, until the scope of each negation is a unique basic formula. The result is transformed into a disjunction of constraints of the form $b_1 \wedge \ldots \wedge b_n \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_m \wedge p$.
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 $n > 1$ is 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.
   (a) 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.
   (b) 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 $\neg \top$.

In the search process for $\mathcal{RH}$-equivalent elemental constraints, there are two ways of eliminating real quantifiers, one of them will simplify 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 $t_1, \ldots, t_l, p_1, \ldots, p_l$.*

[1] Due to lack of space we have not included the corresponding decision procedure.

3.2 Eliminating Quantifiers

The solving procedure for $\mathcal{RH}$-constraints transforms, under certain conditions on the order of the quantified variables, a constraint into an $\mathcal{RH}$-equivalent elemental constraint. It proceeds by eliminating quantifiers of quantified $\mathcal{RH}$-normal forms. Quantifiers over real variables are eliminated using algebraic methods as *CAD*, [3]. The elimination of symbolic quantifiers is based on two rules, $(Pair)$ and $(Elim)$. They transform constraints into $\mathcal{RH}$-equivalent simpler ones that are boolean combinations of basic formulas. The rules $(Pair)$ and $(Elim)$ will be presented together with the theorems that constitute their theoretical foundations. The theorem that is the foundation of the first rule 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 valid for $\mathcal{RH}$-constraints, due to 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 superindex 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 $\vdash_{\mathcal{RH}}$ verifies 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 $\overline{U}$ verifying $F_2\sigma \vdash_{\mathcal{RH}} (\overline{X} \approx \overline{t})\sigma^\star$, 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.*

The second rule in which the main procedure is based requires the following definition.

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, 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. *$\overline{W}$.*

This definition depends on the verification of some properties for every solved form of E, which is difficult to check directly. However, it is possible to provide for a characterization of this concept very easy to check. Moreover, 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. $\overline{W}$.

Proposition 3 (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.*

The next theorem will be the core of the transformation rule $(Elim)$, which is the responsible for the elimination of symbolic quantifiers. Its proof requires several technicalities, which are detailed on [7].

Theorem 4 (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 next 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 b have the form $\exists \overline{U}(\overline{X} \approx \overline{t}(\overline{U}, \overline{x}) \wedge Y \approx s(\overline{U}, \overline{x}))$. 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 b' have the form $\exists \overline{U}(\overline{X} \approx \overline{t}(\overline{U}, \overline{x}) \wedge Y \approx s(\overline{U}, \overline{x}))$. 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', with forms $b \equiv \exists \overline{U}, \overline{V}(\overline{X} \approx \overline{t}(\overline{U}, \overline{x}) \wedge Y \approx s(\overline{U}, \overline{V}, \overline{x}))$ 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.*

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 4.*

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

3.3 The Transformation Algorithm

As we have mentioned before, the algorithm to check $\mathcal{RH}$-satisfiability should be classified as a quantifier elimination technique. That referred 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.

1) Producing a constraint in prenex form $\mathcal{RH}$-equivalent to C. In this way, a formula $\mathit{\Pi} C'$ is obtained so that $C \equiv_{\mathcal{RH}} \mathit{\Pi} C'$, where $\mathit{\Pi}$ is a sequence of quantifiers and C' is quantifier-free.
2) Transforming $\mathit{\Pi} C'$ into $\mathit{\Pi}(C_1 \vee \ldots \vee C_n)$, where, for each $1 \leq i \leq n$, C_i is a conjunction of equalities, disequalities and polynomials.
3) Transforming $\mathit{\Pi}(C_1 \vee \ldots \vee C_n)$ into an $\mathcal{RH}$-equivalent quantified $\mathcal{RH}$-normal form, modifying each C_i, $1 \leq i \leq n$ as follows.
   3.1) Let E be a solved form for the system of equalities and polynomial in C. C_i is removed from the disjunction if such solved form E does not exist. Otherwise, let $\overline{W}$ be the set of parameters in E, free in C_i, and let $\overline{W'}$ be a set of variables not occurring in E, in bijection with $\overline{W}$. Now, replace, in the system E, each variable $W \in \overline{W}$ with the corresponding $W' \in \overline{W'}$. Later, augment the result with the equalities $\overline{W} \approx \overline{W'}$. Finally, quantify existentially the conjunction of the equalities over all its parameters, leaving the polynomial unaffected, therefore rendering a constraint of the form $p \wedge b$ ($b \equiv \top$ if there are no equalities in C_i).
   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, to $p \wedge b$.
   3.3) At this stage, C_i has been removed or transformed to some $p \wedge b \wedge \neg(p_1 \wedge b_1) \wedge \ldots \wedge \neg(p_s \wedge b_s)$, $s \geq 0$, which is finally $\mathcal{RH}$-normalized.

The result of this transformation is a constraint $\mathit{\Pi} C''$ in quantified $\mathcal{RH}$-normal form, where $C'' \equiv D_1 \vee \ldots \vee D_m$ and each D_i is of the form $p' \wedge b \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_k$.

Phase II: Quantifier elimination algorithm.

4) Eliminating the innermost quantifier.
   Let $\mathit{\Pi} = \mathit{\Pi}' Q\, \nu$, where $Q \in \{\exists, \forall\}$ and $\nu \in \mathcal{V}$. We intend to eliminate the quantification $Q\, \nu$, obtaining an $\mathcal{RH}$-normal form $\mathcal{RH}$-equivalent to $Q\, \nu C''$. If $\nu : h$, the algorithm can perform this task for any constraint. However, if $\nu : r$, it may or may not be able to do it, depending on C''. If $Q\, \nu$ cannot be eliminated by the following criteria, but $\mathit{\Pi}' = \mathit{\Pi}'' Q\, \nu_1 \ldots Q\, \nu_l$, $l \geq 1$, and for some i, $1 \leq i \leq l$, $Q\, \nu_i$ does verify some of those criteria, then move $Q\, \nu_i$ to the right of $Q\, \nu$, and proceed. If no such quantifier is found, stop.

4.1) $\nu = x : r$, $Q \equiv \exists$. In this case, $Q\,\nu C'' \equiv \exists x\,(D_1 \vee \ldots \vee D_m) \equiv_{\mathcal{RH}} \exists x\,D_1 \vee \ldots \vee \exists x\,D_m$. Now, it is required that for each D_i, one of the next two conditions holds, otherwise, the algorithm stops without success.
   (a) x occurs, at most, in p'. Therefore, $\exists x\,D_i \equiv \exists x\,p' \wedge b \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_k$. $\exists x\,p'$ can be replaced by an $\mathcal{RH}$-equivalent, quantifier-free Tarski formula, obtained by applying a *CAD*, then producing the elimination of the quantification $\exists x$.
   (b) x occurs also in $b \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_k$, and a polynomial of the form $(x \approx t_1 \wedge q_1) \vee \ldots \vee (x \approx t_l \wedge q_l)$ solved for x, $\mathcal{RH}$-equivalent to p', can be found. Then, $\exists x D_i$ is replaced by the $\mathcal{RH}$-equivalent constraint $(q_1 \wedge (b \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_k)[t_1/x]) \vee \ldots \vee (q_l \wedge (b \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_k)[t_l/x])$.

4.2) $\nu = X : h$, $Q \equiv \exists$. Introducing the existential quantifier over X in the disjunction, $\exists X\,D_1 \vee \ldots \vee \exists X D_m$ is obtained. The aim of this step is to replace every $\exists X\,D_i$, $1 \leq i \leq m$, by a boolean combination of basic formulas and polynomials. Let $D''_i \equiv b \wedge \neg b'_1 \wedge \ldots \wedge \neg b'_k$, $1 \leq i \leq m$, and thus $\exists X\,D_i \equiv_{\mathcal{RH}} p' \wedge \exists X\,D''_i$. Now, if $k = 0$, $\exists X\,D''_i$ can be easily transformed into a basic formula, otherwise the rule $(Pair)$ must be applied, replacing $\exists X\,D''_i$ by $\exists X\,(b \wedge \neg b'_1) \wedge \ldots \wedge \exists X\,(b \wedge \neg b'_k)$. Each member of this conjunction is further transformed by means of the rule $(Elim)$, rendering a boolean combination of polynomials and basic formulas, therefore eliminating the quantifier over X.

4.3) $Q \equiv \forall$. $Q\,\nu C''$ is $\mathcal{RH}$-equivalent to $\neg\exists\nu\,\neg C''$. Now, $\neg C''$ is $\mathcal{RH}$-normalized, producing $D' \equiv D'_1 \vee \ldots \vee D'_u$. Next, the quantifier $\exists\nu$ is eliminated, following 4.1) or 4.2), depending on the type of ν.

5) The constraint produced is a boolean combination of polynomials and basic formulas, which must be $\mathcal{RH}$-normalized. We denote the resulting constraint by C'''. Since $Q\,\nu C'' \equiv_{\mathcal{RH}} C'''$, $\Pi C''$ has been replaced by the constraint $\Pi' C'''$, therefore removing the innermost quantifier.

6) If Π' is empty, stop. Otherwise, go on processing $\Pi' C'''$ starting at step 4).

It must be noticed that termination of the transformation algorithm is guaranteed, because the length of the quantifier prefix decreases on every loop. It will successfully finish when the obtained constraint is elemental.

The algorithm just described constitutes the basis of the constraint solver. The next phase of the solving procedure should be to check the $\mathcal{RH}$-satisfiability of the output of the transformation. When such tansformation is successful, the checking will be carried out by the corresponding decision algorithm dealing with elemental constraints.

4 Examples

The aim of this section is just to show, using examples, the mechanism of the transformation algorithm. In order to find examples of programs of *HH($\mathcal{RH}$)*, see [13]. There it is also described the goal solving procedure, which will call the constraint solver to check $\mathcal{RH}$-satisfiability of answer constraints. More examples illustrating the expressivity of *HH($\mathcal{RH}$)* can be found in [7].

Example 1 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:

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

According to 4.3), the quantification $\forall Z$ is transformed into $\exists Z$, and then $\mathcal{RH}$-normalization is carried out, rendering:

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

$\exists Z$ 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 to $\exists Z_1 (X \approx Z_1 \wedge Y \approx Z_1)$ according to $4.2)$ $(k = 0)$.

Applying $5)$, the $\mathcal{RH}$-normalization yields:

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

Step $4.2)$ is applied after moving $\exists X$ to the right, and 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, carrying out step 5), the quantified $\mathcal{RH}$-normal form below is obtained:

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

Following 4.1), 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, step 4.1.(a) can be now applied, transforming $\exists y (x + y \approx 1)$ into $\top$. With regard to the second one, following 4.1.(a) again, $\exists y (x + y \approx 1 \wedge \neg(x \approx y))$ is transformed into $\neg(x \approx 1/2)$. Therefore,

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

The $\mathcal{RH}$-satisfiability of this elemental constraint can be proved.

Example 2 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, we will use the usual notation of PROLOG for them. 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:

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

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.

Now, according to 4.3), after the implicit $\mathcal{RH}$-normalization and the distribution of the existential quantification $\exists x_3$ the current constraint is:

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

The quantification $\exists x_3$ is trivially removed in the first two cases. 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, steps 4.1.(b), followed by 5). are carried out, rendering:

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

At this stage, in order to eliminate the quantifier $\exists L$, step 4.2) 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

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

Finally, using $4.1(a)$, the current constraint is converted to:

$$C_2 \equiv_{\mathcal{RH}} \neg x_1 \approx 1 \wedge \neg x_1 \approx -1.$$

5 Conclusion

The constraint system $\mathcal{RH}$ has been defined joining the axiomatization of the algebra of finite trees together with the axiomatization of the field of real numbers. Both theories are decidable, and our interest is to find a decision procedure for the combination of both theories. 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 remarkable that the latter applies only to quantifier-free formulas, and the technique of propagation of equalities on which it leans is not useful here when quanifier elimination is carried ot, since this propagation incorporates equalities implied from the polynomial, but does not replace it, which does not help to the elimination of a quantifier. However, it could be used to check $\mathcal{RH}$-satisfiability of constraints with no mixed quantifier prefix.

Starting from 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 been found. This procedure can be considered 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. So, it has been defined as simple as possible to facilitate this study. However, regarding implementation, many refinements and improvements, as the incorporation of heuristics, would be necessary. With respect to the efficiency, the algorithm is very naïve, since it incorporates parts like the *Solve-Tree* algorithm, which is exponential. The tasks concerning elimination of real quantifiers computing a *CAD* have polynomial complexity [3], but they can be reused when the following real quantifier is eliminated. Another desirable feature concerning to 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, as well as to improve the efficiency of our procedure, in order to implement it as a proper $\mathcal{RH}$-constraint solver.

Acknowledgements: We are grateful to Jacobo Torán, Javier Leach and Jesús Escribano for their collaboration during the first stages of the development of this work.

References

1. Buchberger, B. and , Winkler, F. (eds.) *Gröebner Bases and Applications*, Cambridge Univ. Press, 1998.
2. Caprotti, O. *Extending RISC-CLP($\mathcal{R}eal$) 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*, Technical Report 01-121, Dep. of Computer Science, Univ. Complutense of Madrid, 2001.
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($\mathcal{R}$) 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.