Equational Constraint Solving using Quasi-solved Forms*

Javier Álvez and Paqui Lucio

Dpto de Lenguajes y Sistemas Informticos, Facultad de Informática,
Universidad del País Vasco, Paseo Manuel de Lardizabal, 1,
Apdo 649, 20080-San Sebastián, Spain.
jibalgij,jiplucap@si.ehu.es

Abstract. In this paper, we present a deterministic syntactic method for solving general equational constraints. Our method is based on a class of equational formulas, called *answers*, which allows restricted universal quantification and for which the satisfiability test is not hard but not trivial. Thus, we consider it as a quasi-solved form. The procedure keeps, at each step of quantifier elimination, the inner formula being a disjunction of answers. For that, we only need two basic operations - negation and conjunction - on answers which can be efficiently implemented. With respect to purely existential solved forms, answers provide two main advantages for efficiency purposes: (1) they are weaker restricted, thus each quantifier elimination step requires less transformational work and (2) they are more compact - or represent greater sets of solutions - so, each step deals with a smaller quantity of disjuncts.

1 Introduction

A *constraint* is a logical formula that represents a (possibly infinite) collection of data which are its solutions. In fact, a *solution* of a constraint is an assignment of domain values to its free variables that satisfies the constraint. That leads us to consider a domain for constraint interpretation. A *constraint solving method (or algorithm)* takes as input a constraint and produces the set of all its solutions or, more precisely, some particular representation of it. Syntactic methods are rewriting process that transform the input constraint into a disjunction of constraints in the so-called *solved form* which represents its solutions. Hence, solved form is a key notion for any syntactic constraint solving method. Following [5] and [9], solved forms are supposed to have the following three properties:

1. *Solvability*: Every solved form not identical to the constant False has at least one solution
2. *Simplicity*: Every solution can be easily obtained from a solved form
3. *Completeness*: Every constraint is equivalent to a finite disjunction of solved forms.

* This work has been partially supported by the Spanish Project TIC 2001-2476-C03.

It is worthy to notice that 1 and 3 depend on the considered interpretation domain and also that "easily" is a vague point in 2.

An *equational constraint* is an arbitrary first-order formula built over a signature of function symbols and equality as unique predicate symbol. Equational constraints are interpreted over term algebras. Equational solving is useful in so many areas of computer science that it is difficult to enumerate them. In particular, there are some special cases − such as unification, term complement, word problems, etc − which are of great interest and have been extensively studied (e.g. [3, 8, 11, 13]).

A *unification constraint* is an equational constraint which does neither contain negation nor universal quantification. It is well-known that every unification constraint can be reduced to a finite disjunction of most general unifiers. This is the standard solved form notion in unification constraint solving. A most general unifier is an idempotent substitution $\bigwedge_{i=1}^{n}(x_i = t_i)$ of the variables x_i (each appears once) by the terms t_i (whose variables are considered to be existential).

Universal quantification and negation arise naturally in many applications of equational reasoning, such as inductive theorem proving, using counter-examples in learning and theorem proving, normal logic programming, etc. The general validity problem of equational formulas in term algebras has been proved to be decidable by different methods and for different classes of term algebras (cf. [8, 10, 12, 15–18, 21]). In equational constraint solving, negation is unavoidable. For example, the constraint $\forall v(x \neq f(v, v))$ can not be finitely represented without disequations. Hence substitutions (or mgu's) do not serve as solved forms. Many effort has been dedicated to study the transformation of (subclasses of) equational formulae into the so-called "equations only" solved forms. It has been proved (in [19]) that whenever an equational formula is equivalent to a finite set of unifiers, then they can be computed. There are solving methods (cf. [6, 7, 14]) for reducing certain subclasses of equational constraints into (a disjunction of) "only equations" solved forms. These methods fail when the input constraint can not be expressed without negation.

The syntactic methods for total solving of general equational constraints (such as [8, 15, 16]) require a more general notion of solved form, in which negation should be allowed. On the contrary, universal quantification can be dropped from any equational formula (see [8]) on the basis of the *quantifier elimination technique*. Solving methods usually work by quantifier elimination. In order to erase all the universal quantifiers of the original constraint, some other existential quantifiers must also be eliminated. Let us remind the essence of this early technique of symbolic logic. Initially, the input constraint is transformed into its prenex form $\langle Q_i^* \overline{u}^i \rangle_{i=1}^{n}(\varphi)$ where $Q_i^* \in \{\forall^*, \exists^*\}$ is a block of identical quantifiers on a tuple of variables $\overline{u_i}$ and φ is a quantifier-free equational constraint. Then, the technique consists on iterate a basic procedure that removes the innermost block of quantifiers. In this iteration, each step alternates the (universal/existential) character of the quantifier to be eliminated. However, by the double negation equivalences ($\forall^* \overline{u}(\varphi) \equiv \neg\exists^* \overline{u}(\neg\varphi)$ and $\exists^* \overline{u}(\varphi) \equiv \neg\forall^* \overline{u}(\neg\varphi)$), it is enough to implement a basic procedure for one concrete quantifier. Let us

suppose that the basic procedure eliminates the innermost existential [1] block of quantifiers. That is, it transforms any formula $\exists^* \overline{v}(\varphi)$ into a finite disjunction $\bigvee_{j=1}^{k} \psi_j$ of solved forms. As a result, a prenex formula whose prefix is ended by an existential block $\langle Q_i^* \overline{u}^i \rangle_{i=1}^m \exists^* \overline{v}_{m+1}(\varphi)$ is rewritten as

$$\langle Q_i^* \overline{u}^i \rangle_{i=1}^m \bigvee_{j=1}^{k} \psi_j \text{ where each } \psi_j \text{ is in solved form.} \tag{1}$$

Whereas, if the last block is universal, then double negation is applied before the subformula $\exists^* \overline{v}^{m+1} \neg(\varphi)$ is converted into a disjunction of solved forms:

$$\begin{aligned} \langle Q_i^* \overline{u}^i \rangle_{i=1}^m \forall^* \overline{v}^{m+1}(\varphi) &\longmapsto \langle Q_i^* \overline{u}^i \rangle_{i=1}^m \neg \exists^* \overline{v}^{m+1} \neg(\varphi) \\ &\longmapsto \langle Q_i^* \overline{u}^i \rangle_{i=1}^m \neg \bigvee_{j=1}^{k} \psi_j \end{aligned} \tag{2}$$

It is clear that the chosen notion of solved form hardly binds the efficiency of this technique. In practice, we must take care over two dependent aspects. First, the number of disjuncts (k in (1) and (2)) handled at each step. Notice that every two steps the disjunction is negated (see (2)). Then, the resulting formula has to be adapted for the next step, by applying the distributive law. Second, the amount of transformational work that is necessary to produce solved forms (from general constraints). This is related to the syntactical restrictions that define a solved form. The more one restricts, the more one transforms, although the commitments of solvability and simplicity should not be forgotten. For a survey of equational solving methods based on different solved form notions the reader is referred to [5]. Usually, solved form notions are existential formulas (as in [8], see Fig. 1) or boolean combinations of existential formulas (cf. [15]).

$$\exists x'_1 \ldots \exists x'_k (x_1 = t_1 \wedge \ldots x_n = t_n \wedge x'_1 \neq s_1 \wedge \ldots \wedge x'_m \neq s_m)$$
where • each variable x_i occurs once, and
• each term s_i is distinct from the variable x'_i

Fig. 1. Basic Formulas ([8])

The satisfiability test for purely existential solved forms is trivial. In return of this advantage, the solver must remove all universal quantifiers. This often requires to apply the so-called *Explosion Rule* ([8])[2] that we recall in Fig. 2. That means, to substitute a formula by a disjunction of as many formulas as function symbols are in Σ. This increases the number of disjuncts to be treated in later

[1] The dual step can be easily formulated for the universal case. The method of [15] is based on existential elimination, whereas [8] deals with universal elimination.

[2] That is the *Weak Domain Closure Axiom* (*WDCA* for short) in the nomenclature of [15].

steps and, also, increments the transformational work. Our proposal is to use a notion of *quasi-solved form* that represents solutions in a more compact form. It allows some restricted form of universal quantification, which is enough to avoid the above rule (ER). We consider that an *answer* is a quasi-solved form because, in the case of finite signature, an answer could be unsatisfiable, but there is a weak satisfiability test. In particular, this test does not further transform the answer. Besides, an answer is always satisfiable for an infinite signature. The reader is referred to Sect. 3 for definition and examples of answers. We consider answers as a reasonably simple, enough expressive and user-friendly notion[3].

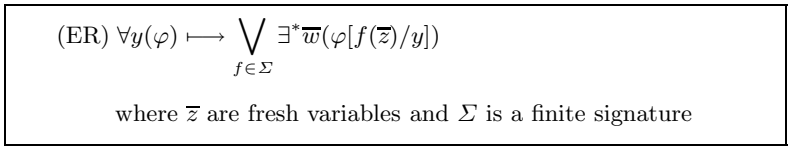

Fig. 2. The Explosion Rule (ER)

Outline of the paper: In the next section we recall some useful definitions and denotational conventions. Section 3 is devoted to the notion of answer. In Sect. 4 we introduce the answer satisfiability test and prove its correctness. In Sect. 5 we present the other two basic operations - conjunction and negation - on answers. We give a summarizing example in Sect. 6. Finally, we resume some conclusions and related work.

2 Definitions and Notation

A term is a variable, or a constant, or a function symbol of arity n applied to n terms. A term is ground if none variable occurs in it. We denote by $Var(t)$ the set of all variables occuring in the term t and $t(\overline{v})$ denotes that $Var(t) \subseteq \overline{v}$. We say Σ-terms to emphasize that the function symbols are taken from a signature Σ and $\mathcal{H}(\Sigma)$ stands for the algebra of all ground Σ-terms or Herbrand universe.

A bar is used to denote tuples of objects, like $\overline{x}$ as abbreviation of the n-tuple of variables $x_1, \ldots, x_n$. Subscripts are used for objects and superscripts for tuples of objects, x_j and $\overline{x}^j$ for example. Concatenation of tuples is denoted by the infix $\cdot$ operator, i.e. $\overline{x} \cdot \overline{y}$ represents the concatenation of $\overline{x}$ and $\overline{y}$.

A Σ-equation is $t_1 = t_2$, where t_1 and t_2 are Σ-terms, whereas $t_1 \neq t_2$ is a Σ-disequation (that abbreviates $\neg(t_1 = t_2)$). By a *collapsing* equation (or disequation) we mean that at least one of its terms is a variable. To avoid confusion, we use the symbol $\equiv$ for the metalanguage equality. We abbreviate collapsing Σ-equation by Σ-`CoEq`.

A *Σ-substitution* σ is a mapping from a finite set of variables $\overline{x}$, called its domain, into a set of Σ-terms. It is assumed that σ behaves as the identity for the variables outside its domain.

[3] Actually, we use answers, *as user-answers*, in a prototypic implementation of constructive negation (cf. [1] and ˜http://www.sc.ehu.es/jiwlucap/BCN.html).

We intentionally confuse a substitution σ with domain $\overline{x}$ with the conjunction of equations $\bigwedge_i x_i = t_i$ where each $t_i \equiv \sigma(x_i)$. Besides, we also write $\varphi[\overline{t}/\overline{x}]$ to denote the simultaneous substitution of each variable x_i by the term t_i in φ.

We say that a term t_1 *subsumes* another term t_2 if there exists a substitution σ such that $\sigma(t1) = t2$.

A Σ-substitution σ such that $\sigma(x_i)$ is ground for each $x_i \in \overline{x}$ is called an *Σ-assignment*.

The *most general unifier* of a set of terms $\{s_1, \ldots, s_n\}$, denoted $mgu(\overline{s})$, is an idempotent substitution σ such that $\sigma(s_i) \equiv \sigma(s_j)$ for all $i, j \in 1..n$ and for any other substitution θ with the same property, $\theta \equiv \sigma' \cdot \sigma$ holds for some substitution σ'. For tuples, $mgu(\overline{s}^1, \ldots, \overline{s}^m)$ is an abbreviation of $\sigma_1 \cdot \ldots \cdot \sigma_n$ where $\sigma_i \equiv mgu(s_i^1, \ldots s_i^m)$ for all $i \in 1..n$.

3 The Notion of Answer

In this section we present the notion of answer and give some illustrative examples. The following definition also introduces some notational conventions on the variables occurring in an answer.

Definition 1. A *Σ-answer for the variables $\overline{x}$* is either a constant (True, False) or a formula of the form $\exists^* \overline{w}(a(\overline{x}, \overline{w}))$ where $a(\overline{x}, \overline{w})$ is a conjunction of both

- Σ-CoEqs of the form $x_i = t_i(\overline{w})$, and
- Σ-UQCDs[4] of the form $\forall^* \overline{v}(w_j \neq s(\overline{w}, \overline{v}))$ where w_j does not occur in the term s and s is not a single universal variable in $\overline{v}$.

where each x_i occurs at most once and every left-hand side variable w_j of a UQCDs occurs in at least one term $t_i(\overline{w})$. ∎

Notice that the scope of universal variables is restricted to one single disequation. However, there is no restriction about repetition of existential, neither universal, variables. We use a Prolog-like notation to write answers in the computer. Traditional Prolog-variables $\_\langle char \rangle$ are used to represent existential variables and for universal ones we introduce new variables of the form $*\langle char \rangle$. For instance,

$$\exists w_1 \exists w_2 (x = f(w_1, w_2) \wedge w_1 \neq g(w_2) \wedge \forall v_1 (w_2 \neq f(w_1, v_1)))$$

is an answer for the variable x which is written as follows:

$$\mathtt{x} = \mathtt{f}(\mathtt{\_A}, \mathtt{\_B}),\ \mathtt{\_A} \neq \mathtt{g}(\mathtt{\_B}),\ \mathtt{\_B} \neq \mathtt{f}(\mathtt{\_A}, *\mathtt{D})$$

It is obvious that any answer can be represented in this notation.

The following examples show that answers provide a compact and explanative description of the sets of solutions that they represent.

[4] Σ-UQCD stands for universally quantified collapsing Σ-disequation.

Example 2. Let $\Sigma \equiv \{a/0, g/1, f/2\}$, the Σ-answer:

$$\exists w_1 \exists w_2 (x = f(w_1, w_2) \wedge w_1 \neq g(w_2) \wedge \forall v_1 (w_2 \neq f(w_1, v_1)))$$

is equivalent to the disjunction of the following eight basic formulas:

1. $x = f(a, a)$
2. $\exists w_1 \exists w_2 (x = f(a, f(w_1, w_2)) \wedge w_1 \neq a)$
3. $\exists w_1 (x = f(a, g(w_1)))$
4. $\exists w_1 \exists w_2 (x = f(f(w_1, w_2), a))$
5. $\exists w_1 \exists w_2 \exists w_3 \exists w_4 (x = f(f(w_1, w_2), f(w_3, w_4)) \wedge w_3 \neq f(w_1, w_2))$
6. $\exists w_1 \exists w_2 \exists w_3 (x = f(f(w_1, w_2), g(w_3)))$
7. $\exists w_1 (x = f(g(w_1), a) \wedge w_1 \neq a)$
8. $\exists w_1 \exists w_2 \exists w_3 (x = f(g(w_1), f(w_2, w_3)) \wedge w_2 \neq g(w_1))$ ∎

Example 3. The following equational constraint of signature $\Sigma \equiv \{a/0, g/1, f/2\}$:

$$\exists w_1 \exists w_2 \forall y_1 \forall y_2 (\ f(f(w_1, a), f(w_2, x_2)) \neq f(f(y_1, a), f(y_2, y_2)) \ \wedge \\ f(g(y_2), x_1) \neq f(x_2, f(y_1, y_1))\ )$$

is equivalent to the following two answers:

(A1) $\exists w (x_2 = w \wedge \forall v (w \neq g(v)))$
(A2) $\exists w_1 \exists w_2 (x_1 = w_1 \wedge x_2 = g(w_2) \wedge \forall v (w_1 \neq f(v, v)))$.

which are written as $\mathtt{x_2 = \_A,\ \_A \neq g(*B)}$ and $\mathtt{x_1 = \_A,\ x_2 = g(\_B),\ \_A \neq f(*C, *C)}$. It is also equivalent to the disjunction of the following five basic formulas:

(B1) $x_2 = a$
(B2) $\exists w_1 \exists w_2 (x_2 = f(w_1, w_2))$
(B3) $\exists w (x_1 = a \wedge x_2 = g(w))$
(B4) $\exists w_1 \exists w_2 (x_1 = g(w_1) \wedge x_2 = g(w_2))$
(B5) $\exists w_1 \exists w_2 \exists w_3 (x_1 = f(w_1, w_2) \wedge x_2 = g(w_3) \wedge w_1 \neq w_2)$

It is easy to realize that the answer (A1) is equivalent to (B1)∨(B2) and the answer (A2) is equivalent to (B3)∨(B4)∨(B5). ∎

4 The Satisfiability Test

We deal with the satisfiability of answers in the domain $\mathcal{H}(\Sigma)$ for some Σ that can be finite or infinite. It is easy to realize that a conjunction of CoEq of the form $\overline{x} = \overline{t}(\overline{w})$ where each $x_i \in \overline{x}$ occurs at most once is always satisfiable in $\mathcal{H}(\Sigma)$. In fact, it is a substitution. With regard to the disequational part of an answer, if Σ is infinite, then any finite conjunction of UQCDs is satisfiable in $\mathcal{H}(\Sigma)$. This is due to the fact that, having infinite many symbols, it always remain infinite possible assignment values for each variable $w_i \in \overline{w}$ that keep satisfiable a given finite set of UQCDs. On the contrary, if Σ is finite, the disequational part of an answer is not necessarily satisfiable in $\mathcal{H}(\Sigma)$.

Example 4. The following two conjunctions of Σ-UQCDs are unsatisfiable in $\mathcal{H}(\Sigma)$ (for $\Sigma \equiv \{a/0, g/1, f/2\}$):

1. $w \neq a \wedge \forall v(w \neq g(v)) \wedge \forall v_1 \forall v_2(w \neq f(v_1, v_2))$
2. $w \neq a \ \wedge \ w \neq g(a) \ \wedge \ \forall v(w \neq g(g(v))) \ \wedge$
   $\forall v_1 \forall v_2(w \neq g(f(v_1, v_2))) \ \wedge \ \forall v_1 \forall v_2(w \neq f(v_1, v_2))$ ∎

Notice that $\mathcal{H}(\Sigma)$, for finite Σ, can be finite or infinite. The most interesting case is when $\mathcal{H}(\Sigma)$ is infinite and Σ is finite.

An answer $\exists^* \overline{w}(a(\overline{x}, \overline{w}))$ is satisfiable in $\mathcal{H}(\Sigma)$ if and only if there is at least one Σ-assignment with domain $\overline{w}$ that makes true the conjunction of UQCDs inside $a(\overline{x}, \overline{w})$. Figure 3 outlines our answer satisfiability test for a finite signature Σ and a finite conjunction of UQCDs. For exposition purposes, we represent the domain $Dom(w_i)$ of the possible assignment values for the variable w_i by a *domain formula* denoted by $\Delta(w_i)$. In other words

$$Dom(w_i) \equiv \{t \in \mathcal{H}(\Sigma) \mid \mathcal{H}(\Sigma) \models \Delta[t/w_i]\}$$

Initially, each $w_i \in \overline{w}$ can take any value in $\mathcal{H}(\Sigma)$. For example, the initial domain of each $w_i \in \overline{w}$ for $\Sigma \equiv \{a/0, g/1, f/2\}$ is given by

$$\Delta(w_i) \equiv w_i = a \vee \exists u(w_i = g(u)) \vee \exists u_1 \exists u_2(w_i = f(u_1, u_2))$$

For the lack of efficiency, our answer satisfiability test does not exhaustively obtain the domain $Dom(w_i)$ of each $w_i \in \overline{w}$. Instead of that, it works by value elimination in two step. The basic idea is that a UQCD disallows some values in $Dom(w_i)$. Each step takes into account a different (syntactic) subclass of UQCDs. The first step treats the UQCDs without existential variables (that is $\overline{w} \equiv \emptyset$) and without repetitions in their universal variables $\overline{v}$. Roughly speaking, this is the only subclass of UQCDs that can transform an infinite domain into a finite one. Let us give an example of this process for eliminating values from the initial domains that is used at the first step and outlined in Fig. 4.

Example 5. Let $\forall v_1 \forall v_2(w_1 \neq f(a, f(v_1, v_2)))$ be the input UQCD and let

$$\Delta(w_1) \equiv w_1 = a \vee \exists u(w_1 = g(u)) \vee \exists u_1 \exists u_2(w_1 = f(u_1, u_2))$$

be the initial domain formula. Since $f(a, f(v_1, v_2))$ does neither unify with a nor $g(u)$, therefore both terms remain in the domain $Dom(w_i)$ (in other words, the already treated part of $\Delta(w_1)$ is $w_1 = a \vee \exists u(w_1 = g(u)))$. Now, $f(a, f(v_1, v_2))$ does not subsume $f(u_1, u_2)$, so we unfold it in u_1. Hence, the not yet treated part of the domain formula $\Delta(w_1)$ becomes

$$\exists u_2(w_1 = f(a, u_2)) \vee \exists u_1 \exists u_2(w_1 = f(g(u_1), u_2)) \vee \exists \overline{u}(w_1 = f(f(u_1, u_3), u_2))$$

Again $f(a, f(v_1, v_2))$ does not subsume $f(a, u_2)$. Therefore, we unfold it in u_2

$$w_1 = f(a, a) \vee \exists u_2(w_1 = f(a, g(u_2))) \vee \exists \overline{u}(w_1 = f(a, f(u_2, u_3))) \vee$$
$$\exists u_1 \exists u_2(w_1 = f(g(u_1), u_2)) \vee \exists \overline{u}(w_1 = f(f(u_1, u_3), u_2))$$

Input: $\Sigma \equiv \{f_1\backslash a_1, \ldots, f_k\backslash a_k\}$ $--$ signature: function-symbol\arity

$\exists^* \overline{w}(\bigwedge \forall^* \overline{v}(w_i \neq s(\overline{w}, \overline{v})))$ $--$ disequational part of a Σ-answer

Precondition: For all $w_i \in \overline{w}$: $\Delta(w_i) \equiv \bigvee_{f \in \Sigma} \exists^* \overline{u}(w_i = f(\overline{u}))$
where each u_i appears once.

First Step: Eliminate all values in $Dom(w_i)$ disallowed by the UQCDs of the form: $\forall^* \overline{v}(w_i \neq s(\overline{v}))$ where each v_k occurs at most once in $s(\overline{v})$ (according to the algorithm of Fig 4)

If $Dom(w_i) \equiv \emptyset$ for some i
then the answer is **unsatisfiable**
else let $Fin(\overline{w}) \equiv \{w_i | Dom(w_i)$ is finite$\}$
If $Fin(\overline{w}) \equiv \emptyset$
then the answer is **satisfiable**
else go to Second step

Second Step: Search an assignment σ such that $\sigma(w_i) \in Dom(w_i)$ for each $w_i \in Fin(\overline{w})$ and σ satisfies all UQCDs of the form: $\forall^* \overline{v}(w_i \neq s(\overline{w}, \overline{v}))$ where $w_i \in Fin(\overline{w})$ and $Var(s) \subseteq Fin(\overline{w}) \cup \overline{v}$

If such σ exists **then** the answer is **satisfiable**
else the answer is **unsatisfiable**

Fig. 3. Answer Satisfiability Test

Finally, $f(a, f(v_1, v_2))$ subsumes $f(a, f(u_2, u_3))$ and does not unify with none of the other four terms. Therefore, the resulting domain is given by

$$w_1 = a \vee \exists u(w_1 = g(u)) \vee w_1 = f(a, a) \vee \exists u_2(w_1 = f(a, g(u_2))) \vee$$
$$\exists u_1 \exists u_2(w_1 = f(g(u_1), u_2)) \vee \exists \overline{u}(w_1 = f(f(u_1, u_3), u_2)) \quad \blacksquare$$

After the elimination algorithm of Fig. 4, each $Dom(w_i)$ can be either empty or a finite set of ground terms or infinite. The last case is expressed by a $\Delta(w_i)$ with at least one auxiliary existential variable. Then, it decides that the input answer is unsatisfiable when some $Dom(w_i) \equiv \emptyset$. By the contrary, the answer is satisfiable whether it remains infinite values in all $Dom(w_i)$. The first step is very often enough to decide.

Example 6. The answer of Example 2 and also (A1) and (A2) of Example 3 are decided to be satisfiable at the first step, since $Fin(\overline{w}) \equiv \emptyset$. Only in the case of (A1) one UQCD is treated. In the other two cases, no UQCD satisfies the condition. Moreover, the unsatisfiability of both constraints in Example 4 is also decided at the first step. $\blacksquare$

By the contrary, if no $Dom(w_i)$ is empty and at least one $Dom(w_i)$ is finite, then the procedure looks for the existence of an assignment σ such that

$\sigma(w_i) \in Dom(w_i)$ for each $w_i \in Fin(\overline{w})$. The remaining variables are not taken into account because (as we will prove) they are irrelevant for the satisfiability decision. Moreover, the assignment σ must satisfy only one concrete subclass of the remaining UQCDs: the ones whose existential variables are included in $Fin(\overline{w})$. The existence of such σ decides the satisfiability of the input answer.

Input: Σ −− Signature
$Dom(w_i)$ −− Domain
$\forall^* \overline{v}(w_i \neq s(\overline{v}))$ −− Σ-UQCD s. t. each v_k occurs at most once in $s(\overline{v})$
For each $t_j \in Dom(w_i)$ **do**
  If $mgu(s(\overline{v}), t_j)$ doesn't exist
    then t_j remains in $Dom(w_i)$
    elsif $s(\overline{v})$ subsumes t_j
      then t_j is removed from $Dom(\overline{w}_i)$
      else Refine $Dom(w_i)$ by unfolding the term t_j (according to Σ) at one of its existential variables.

Fig. 4. Elimination Algorithm (First Step of Fig. 3)

In order to prove the correctness of the satisfiability test, the only crucial matter is that all the rejected disequations are really irrelevant for deciding the satisfiability of the input answer. Along the test, we discard all the UQCDs $\forall^* \bar{v}(w_i \neq s(\overline{w}, \overline{v}))$ which satisfy the following two properties:

(P1) $Var(s) \cap \overline{w} \not\equiv \emptyset$ or some v_k appears more than once in s, and
(P2) At least one existential variable w_j is not in $Fin(\overline{w})$.

We will show that these UQCDs do not matter for the decision. Our proof is based on the fact that the domain of each variable $w_j \in (\overline{w} \setminus Fin(\overline{w}))$ is, along the whole test, defined by a *uniform* domain formula $\Delta(w_j)$.

Definition 7. We say that $\Delta(w_j) \equiv \bigvee_k \exists^* \overline{u}(w_j = t_{j,k}(\overline{u})))$ is *uniform* if and only if each u_i occurs at most once in $t_{j,k}(\overline{u})$. ∎

The following auxiliary result will allow us to extend any assignment σ with domain $Fin(\overline{w})$ to the variables $\overline{w} \setminus Fin(\overline{w})$, giving an assignment σ' that satisfies every uniform domain formula $\Delta(w_i)$ for $w_i \in \overline{w} \setminus Fin(\overline{w})$.

Proposition 8. *Let $\varphi(\overline{w})$ be a finite conjunction of UQCDs that satisfy the above properties (P1) and (P2). If $\Delta(w_j)$ is uniform for each variable $w_j \in \overline{w} \setminus Fin(\overline{w})$, then any assignment σ with domain $Fin(\overline{w})$ can be extended to an assignment σ' with domain $\overline{w}$ that satisfies:*

$$\sigma \quad \wedge \quad \varphi(\overline{w}) \quad \wedge \bigwedge_{w_j \in \overline{w} \setminus Fin(\overline{w})} \Delta(w_j)$$

Proof. Since $\sigma(w_i)$ is a ground term $t_i \in \mathcal{H}(\Sigma)$ for each $w_i \in Fin(\overline{w})$, we can substitute in $\varphi(\overline{w})$ each $w_i \in Fin(\overline{w})$ for its value t_i. Then, we obtain an equivalent equational formula $\varphi'(\overline{w} \setminus Fin(\overline{w}))$ which is a finite conjunction of UQCDs of one of the following two types:

(T1) $\forall^* \overline{v}(w_k \neq s(\overline{v}))$ where at least one v_j occurs repeatedly in s

(T2) $\forall^* \overline{v}(tw \neq s(\overline{w}, \overline{v}))$ where there is at least one w_j in s and tw is used to denote that it is either a ground term t_i or a variable w_k.

Now, σ' should extend σ by assigning a value in $\mathcal{H}(\Sigma)$ to each $w_k \in \overline{w} \setminus Fin(\overline{w})$ for satisfying:

$$\varphi'(\overline{w} \setminus Fin(\overline{w})) \ \wedge \bigwedge_{w_j \in \overline{w} \setminus Fin(\overline{w})} \Delta(w_j)$$

Indeed there are infinite possibilities for such assignment σ'. By the uniformity of each $\Delta(w_j)$, its conjunction with any finite collection of formulas of type (T1) preserves the infinity of $Dom(w_j)$ (although they also disallow infinite values) for each w_j. Therefore, there are infinite possible tuples of values which can be assigned to the tuple of variables in $\overline{w} \setminus Fin(\overline{w})$ for satisfying the finite conjunction of UQCDs of type (T2). ∎

Theorem 9. *The satisfiability test given by Figures 3 and 4 is correct.*

Proof. Initially, $\Delta(w_i)$ is uniform (in the sense of Def. 7) for each w_i. It is easy to see that uniformity is preserved along the first step, because the UQCDs used to refine the domains have neither existential variables nor repeated universal variables. Therefore, on the basis of Proposition 8, if the second step is able to find an assignment σ, then σ can be extended for satisfying the whole answer. Notice that σ is empty when such decision is taken at the first step, and the same reasoning can be applied. Otherwise, if the second step says that the answer is unsatisfiable after checking a subset of UQCDs, then the whole answer is necessarily unsatisfiable. ∎

5 Basic Operations on Answers

In this section. we explain how the solving method transforms any equational constraint into a disjunction of answers which are filtered by the satisfiability test. As we explain in Sect. 1, we use the quantifier elimination technique by keeping, at each step, the inner formula as a disjunction of answer. Hence, we must transform the inner formula of (2) (in Sect. 1) into a disjunction of answers. To do that, in addition of distribution, we use only two basic operations on answers (below denoted **(1)** and **(2)**):

$$\neg \bigvee_{j=1}^{k} \psi_j \longmapsto \bigwedge_{j=1}^{k} \neg\psi_j \longmapsto^{(1)} \bigwedge_{j=1}^{k} \bigvee_{r=1}^{m_j} \varphi_j^r \longmapsto \bigvee_{r} \bigwedge_{j=1}^{k} \varphi_j^r \longmapsto^{(2)} \bigvee_{r} \bigvee_{h} \gamma_h^r$$

(1) The negation of an answer ψ_j is reduced to a disjunction $\bigvee_{r=1}^{m_j} \varphi_j^r$ of answers.

(2) The conjunction $\bigwedge_{j=1}^{k} \varphi_j^r$ of k answers is reduced to a disjunction $\bigvee_h \gamma_h^r$ of answers.

Now, let us explain that these two operations - negation and conjunction - can be efficiently performed. To do that, we use an auxiliary transformation rule (UD) of Fig. 5 which allows to separate a conjunction of equations in a negated existential formula. We first prove the correctness of the rule (UD).

$$(\mathrm{UD})\ \neg\exists^*\overline{v}[\overline{x} = \overline{t}(\overline{w},\overline{v}) \wedge \varphi] \longmapsto \neg\exists^*\overline{v}^1[\overline{x} = \overline{t}(\overline{w},\overline{v}^1)] \ \vee\ \exists^*\overline{v}^1[\overline{x} = \overline{t}(\overline{w},\overline{v}^1) \wedge \neg\exists^*\overline{v}^2\varphi]$$
$$\text{where } \overline{v}^1 \equiv free(\overline{t}) \cap \overline{v} \text{ and } \overline{v}^2 \equiv \overline{v}\backslash\overline{v}^1$$

Fig. 5. Transformation Rule (UD)

Proposition 10. *The transformation rule (UD) of Figure 5 is correct.*

Proof. The rule (UD) is obtained by successive applications of its simplified version for separating only one equation:

$$(UB)\ \neg\exists^*\overline{v}[x = t(\overline{w},\overline{v}) \wedge \varphi] \longmapsto \neg\exists^*\overline{v}^1[x = t(\overline{w},\overline{v}^1)] \vee \exists^*\overline{v}^1[x = t(\overline{w},\overline{v}^1) \wedge \neg\exists^*\overline{v}^2\varphi]$$

where $\overline{v}^1 \equiv free(t) \cap \overline{v}$ and $\overline{v}^2 \equiv \overline{v}\backslash\overline{v}^1$. To realize this it is enough to see that a formula of the form:

$$\begin{aligned}
&\neg\exists^*\overline{v}^{1,1}[x_1 = t_1(\overline{w},\overline{v}^{1,1})] \quad \vee \\
&\exists^*\overline{v}^{1,1}[x_1 = t_1(\overline{w},\overline{v}^{1,1}) \wedge \neg\exists^*\overline{v}^{1,2}(x_2 = t_2(\overline{w},\overline{v}^{1,2}))] \quad \vee \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \qquad (3)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vee \\
&\exists^*\overline{v}^{1,1}\cdots\overline{v}^{1,n-1}[x_1 = t_1(\overline{w},\overline{v}^{1,1}) \wedge \ldots \wedge x_{n-1} = t_{n-1}(\overline{w},\overline{v}^{1,n-1}) \wedge \\
&\qquad\qquad\qquad\qquad\qquad\qquad \neg\exists^*\overline{v}^{1,n}(x_n = t_n(\overline{w},\overline{v}^{1,n}))]
\end{aligned}$$

is equivalent (in $\mathcal{H}(\Sigma)$) to $\neg\exists^*\overline{v}^1[\overline{x} = \overline{t}(\overline{w},\overline{v}^1)]$ where $\overline{v}^1 \equiv \overline{v}^{1,1}\cdots\overline{v}^{1,n}$. Hence, it suffices to prove that rule (UB) is correct. By conjunction (and distribution) of $\neg\exists^*\overline{v}[x = t(\overline{w},\overline{v}) \wedge \varphi]$ with the tautology

$$\neg\exists^*\overline{v}(x = t(\overline{w},\overline{v})) \vee \exists^*\overline{v}(x = t(\overline{w},\overline{v}))$$

we obtain two disjuncts. The first one is trivially equivalent to $\neg\exists^*\overline{v}(x = t(\overline{w},\overline{v}))$. The second disjunct

$$\neg\exists^*\overline{v}[x = t(\overline{w},\overline{v}) \wedge \varphi] \wedge \exists^*\overline{v}(x = t(\overline{w},\overline{v}))$$

is equivalent to the following adequately renamed constraint:

$$\exists^*\overline{v}^1\forall^*\overline{z}\forall^*\overline{v}^2[x = t(\overline{w},\overline{v}^1) \wedge (x \neq t(\overline{w},\overline{z}) \vee \neg\varphi[\overline{z}\cdot\overline{v}^2/\overline{v}])]$$

By distribution and unification, we obtain a first disjunct $\overline{v}^1 \neq \overline{z}$ which is eliminated after substituting $\overline{z}$ by $\overline{v}^1$ in the second disjunct: $x = t(\overline{w},\overline{z}) \wedge \neg\varphi[\overline{z}\cdot\overline{v}^2/\overline{v}]$.

Hence, we obtain $\exists^*\overline{v}^1\forall^*\overline{v}^2[x = t(\overline{w}, \overline{v}^1) \wedge \neg\varphi[\overline{v}^1 \cdot \overline{v}^2/\overline{v}]]$ which is trivially equivalent to the second disjunct in the right-hand of rule (UB). ∎

Proposition 11. *The negation $\neg\exists^*\overline{w}(a(\overline{x}, \overline{w}))$ of an answer for $\overline{x}$ can be transformed into an equivalent disjunction $\bigvee_j \exists^*\overline{w}(a_j(\overline{x}, \overline{w}^j))$ of answers for $\overline{x}$.*

Proof. We apply the transformation rule (UD) to the negated answer:

$$\neg\exists^*\overline{w}[\overline{x} = \overline{t}(\overline{w}) \wedge \bigwedge \forall^*\overline{v}(w_j \neq s(\overline{w}, \overline{v}))] \longmapsto$$

$$\neg\exists^*\overline{w}[\overline{x} = \overline{t}(\overline{w})] \vee \exists^*\overline{w}[\overline{x} = \overline{t}(\overline{w}) \wedge \bigvee \exists^*\overline{v}(w_j = s(\overline{w}, \overline{v}))]$$

The first disjunct $\neg\exists^*\overline{w}(\overline{x} = \overline{t}(\overline{w}))$ is tranformed into (3) (see the proof of Prop. 5). Then, we replace the variables x_i in the disequations by new variables w'_i, adding the corresponding equation $x_i = w'_i$, to obtain a disjunction of answer for $\overline{x}$. For the second disjunct, it suffices to distribute the internal disjunction into:

$$\bigvee \exists^*\overline{w}[\overline{x} = \overline{t}(\overline{w}) \wedge \exists^*\overline{v}(w_j = s(\overline{w}, \overline{v}))]$$

and then substitute w_j by $s(\overline{w}, \overline{v})$ in $t(\overline{w})$ to yield:

$$\bigvee \exists^*\overline{w}\exists^*\overline{v}(\overline{x} = \overline{t}(\overline{w})[s(\overline{w}, \overline{v})/w_j])$$ ∎

Proposition 12. *The conjunction $\bigwedge_{i=1}^k \exists^*\overline{w}^i(a_i(\overline{x}, \overline{w}^i))$ of k answers for $\overline{x}$ can be transformed into an equivalent disjunction $\bigvee_j \exists^*\overline{w}^j a'_j(\overline{x}, \overline{w}^j)$ of answers for $\overline{x}$.*

Proof. If the $mgu(\overline{t}_1(\overline{w}^1), \ldots, \overline{t}_k(\overline{w}^k))$ does not exist, then the result of the conjunction

$$\bigwedge_{i=1}^{k} \exists^*\overline{w}^i[\overline{x} = \overline{t}^i(\overline{w}^i) \wedge \bigwedge_h \forall^*\overline{v}(w_h^i \neq s(\overline{w}^i, \overline{v}))]$$

is the constant False that can be seen as the empty disjunction. Otherwise, it is a substitution σ. Therefore, we join the equational parts as follows:

$$\exists^*\overline{w}^1 \ldots \exists^*\overline{w}^k[\overline{x} = \sigma(\overline{t}^1(\overline{w}^1)) \wedge \bigwedge_i \bigwedge_h \forall^*\overline{v}(\sigma(w_h^i) \neq \sigma(s(\overline{w}^i, \overline{v})))]$$

That is a constraint of the form:

$$\exists^*\overline{w}[\overline{x} = t'(\overline{w}) \wedge \bigwedge_h \neg\exists^*\overline{v}(r_h(\overline{w}) = s_h(\overline{w}, \overline{v}))]$$

where $\overline{w} \equiv \overline{w}^1 \cdot \ldots \cdot \overline{w}^k$. If $\sigma_h \equiv mgu(r(\overline{w}), s(\overline{w}, \overline{v}))$ does not exist, then the corresponding conjunct is True. Otherwise, since σ_h is an idempotent substitution, $\neg\exists^*\overline{v}(\sigma_h)$ can be transformed into a disjunction $\bigvee_j \exists^*\overline{z}(a_j(\overline{w}, \overline{z}))$ of answers for $\overline{w}$ (similar to (3) in Prop. 5). Hence, the constraint has the form:

$$\exists^*\overline{w}[\overline{x} = t'(\overline{w}) \wedge \bigwedge \bigvee_j \exists^*\overline{z}(a_j(\overline{w}, \overline{z}))]$$

To finish, we apply (in this order) distribution, conjunction of answers, lifting of the internal disjunction and substitution of the equational part of each answer $a_j(\overline{w}, \overline{z})$ in $t'(\overline{w})$. ∎

6 A Complete Example

In this section we demonstrate the application of our deterministic solving method to the following equational constraint with a five-length prefix of quantifier blocks and free variables x_1, x_2, x_3:

$$\begin{array}{ll} \forall y_1(\exists w_1(\forall y_2(& f(x_1, g(y_2)) \neq f(f(w_1, x_2), a) \quad \wedge \\ & w_1 \neq f(y_1, y_1) \quad \wedge \\ & \exists w_2(\forall y_3(f(x_2, a) \neq f(g(y_3), w_1))) \quad \wedge \\ & \forall y_3(\forall y_4(f(x_1, x_2) \neq f(g(x_3), f(y_3, y_4)))) \quad))) \end{array}$$

The preliminary transformation gives the following disjunction of answers for x_1, x_2, x_3, y_1, w_1 prefixed by $\forall y_1 \exists w_1$:

$$\begin{array}{ll} \forall y_1(\exists w_1(& \exists^* \overline{z}(\; x_1 = z_1 \; \wedge \; x_2 = z_2 \; \wedge \; x_3 = z_3 \; \wedge \; y_1 = z_4 \; \wedge \; w_1 = z_5 \; \wedge \\ & \quad z_5 \neq f(z_4, z_4) \; \wedge \; \forall v_1 \forall v_2(z_2 \neq f(v_1, v_2)) \; \wedge \; \forall v(z_2 \neq g(v))) \quad \vee \\ & \exists^* \overline{z}(x_1 = z_1 \; \wedge \; x_2 = z_2 \; \wedge \; x_3 = z_3 \; \wedge \; y_1 = z_4 \; \wedge \; w_1 = z_5 \; \wedge \\ & \quad z_5 \neq f(z_4, z_4) \; \wedge \; z_1 \neq g(z_3) \; \wedge \; \forall v(z_2 \neq g(v))) \quad \vee \\ & \exists^* \overline{z}(x_1 = z_1 \; \wedge \; x_2 = z_2 \; \wedge \; x_3 = z_3 \; \wedge \; y_1 = z_4 \; \wedge \; w_1 = z_5 \; \wedge \\ & \quad z_5 \neq f(z_4, z_4) \; \wedge \; \forall v_1 \forall v_2(z_2 \neq f(v_1, v_2)) \; \wedge \; z_5 \neq a) \quad \vee \\ & \exists^* \overline{z}(x_1 = z_1 \; \wedge \; x_2 = z_2 \; \wedge \; x_3 = z_3 \; \wedge \; y_1 = z_4 \; \wedge \; w_1 = z_5 \; \wedge \\ & \quad z_5 \neq f(z_4, z_4) \; \wedge \; z_1 \neq g(z_3) \; \wedge \; z_5 \neq a) \quad)) \end{array}$$

Notice that the prefix is shorter because answers allow universal quantification and also that they are answers for the free variables of the original constraint and for the variables of the prefix. After that preliminary treatment, quantifier elimination is successively applied until the prefix is erased. The innermost block $\exists w_1$, likewise any existential block, is easily eliminated. In general, we can remove all the CoEqs on the affected variables and the UQCDs whose existential variable has been just removed. In the example, $w_1 = z_5$, $z_5 \neq f(z_4, z_4)$ and $z_5 \neq a$ are erased everywhere. Now, we have a disjunction of answers for x_1, x_2, x_3, y_1 prefixed by $\forall y_1$ which, by double negation, is equivalent to (where $\overline{x} \cdot y_1 = \overline{z}$ abbreviates $x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge y_1 = z_4$):

$$\begin{array}{ll} \neg\exists y_1(& \neg\exists^* \overline{z}(\overline{x} \cdot y_1 = \overline{z} \wedge \forall^* \overline{v}(z_2 \neq f(v_1, v_2)) \wedge \forall v(z_2 \neq g(v))) \quad \wedge \\ & \neg\exists^* \overline{z}(\overline{x} \cdot y_1 = \overline{z} \wedge z_1 \neq g(z_3) \wedge \forall v(z_2 \neq g(v))) \quad \wedge \\ & \neg\exists^* \overline{z}(\overline{x} \cdot y_1 = \overline{z} \wedge \forall^* \overline{v}(z_2 \neq f(v_1, v_2))) \quad \wedge \\ & \neg\exists^* \overline{z}(\overline{x} \cdot y_1 = \overline{z} \wedge z_1 \neq g(z_3)) \quad) \end{array}$$

Then, each negated answer $\neg\exists^* \overline{z}(\ldots)$ produces a disjunction of answers for $\overline{x} \cdot y_1$. By distributing, we obtain a disjunction of conjunctions of answers and, by performing conjunctions, only one conjunction yields a (different from False) answer for $\overline{x} \cdot y_1$:

$$\neg\exists y_1(\exists^* \overline{z}(x_3 = z_1 \wedge x_1 = g(z_1) \wedge x_2 = f(z_2, z_3) \wedge y_1 = z_4))$$

Finally, $\exists y_1$ and $y_1 = z_4$ can be eliminated and the negation of the resulting answer for $\overline{x}$ gives:

$$\begin{array}{ll} \exists^* \overline{z}(x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge \forall^* \overline{v}(z_2 \neq f(v_1, v_2))) & \vee \\ \exists^* \overline{z}(x_1 = g(z_1) \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge z_4 \neq z_1) & \vee \\ \exists^* \overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge \forall v(z_1 \neq g(v))) & \end{array}$$

7 Conclusions and Related Work

We have introduced a new notion of quasi-solved form, called answer, which enables classical quantifier elimination be more efficiently performed for equational constraint solving. Actually, answers allow us to represent sets of solutions in a more compact way and, at the same time, to check answer satisfiability is not very expensive in time. We avoid to yield a huge number of junk formulas by applying a non-hard satisfiability test. Hence, we achieve a trade-off between time and space efficiency. We have presented two additional operations on answers - negation and conjunction - that can be efficiently performed. On the basis of these two operations and the satisfiability test, we have defined a deterministic syntactic method for solving general equational constraints. We have implemented a prototype of this constraint solver in Sictus Prolog v.3.x that is available in ~http://www.sc.ehu.es/jiwlucap/equality_constraints.html. At the same URL, you can also find a CHR version of the answer satisfiability test. To the best of our knowledge, there is not other available implementation of general equational constraint solving.

Regarding to existing equational solved form notions allowing some kind of (restricted) universal quantification, we are aware of the so-called *substitutions with exceptions* (cf. [2]) and also of the notion introduced in [4, 20]. Like our answers, substitutions with exceptions are always satisfiable in the case of infinite signature, but a satisfiability test is needed in the case of finite signature. The main difference between answers and substitutions with exceptions is twofold. First, we allow to mix both kind of variables (universal and existential) in the disequations right-hand terms. Second, we restrict the scope of each universal quantifier to one collapsing disequation, instead of a conjunction of disequations. The latter work ([4, 20]) is centered in the case of infinite signatures. Its main goal is the efficient decidability of equational formulas with a long prefix of quantifiers. Due to that, the solved form notion is not focused on user-friendliness. Actually, it allows unrestricted nesting of negation and quantification.

References

1. J. Álvez, P. Lucio, F. Orejas, E. Pasarella, and E. Pino. Constructive negation by bottom-up computation of literal answers. In *Applied Computing 2004, Proceedings of the 2004 ACM Symposium on Applied Computing*, volume 2, pages 1468–1475, 2004.
2. W. L. Buntine and H.-J. Bürckert. On solving equations and disequations. *Journal of the ACM*, 41(4):591–629, 1994.

3. A. Colmerauer. Equations and inequations on finite and infinite trees. In *2nd International Conference on Fifth Generation Computer Systems*, pages 85–99, 1984.
4. A. Colmerauer and T.-B.-H. Dao. Expresiveness of full first order constraints in the algebra of finite and infinite trees. In *6th Int. Conf. of Principles and Practice of Constraint Programming CP'2000*, volume 1894 of *LNCS*, pages 172–186, 2000.
5. H. Common. Disunification: A survey. In J.L. Lassez and G. Plotkin, editors, *Essays in Honour of Alan Robinson*, 1991.
6. H. Comon. Unification et disunification: Théories et applications. Technical report, PhD thesis, Université de l'Institut Polytechnique de Grenoble, 1988.
7. H. Comon and M. Fernández. Negation elimination in equational formulae. In *Mathematical Foundations of Computer Science*. Springer Verlag, 1992.
8. H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425, 1989.
9. Hubert Comon, Mehmet Dincbas, Jean-Pierre Jouannaud, and Claude Kirchner. A methodological view of constraint solving. *Constraints*, 4(4):337–361, 1999.
10. W. Hodges. Model theory. In *Encyclopedia of Mathematics and its Applications*, volume 42. Cambridge University Press, 1993.
11. J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In G. D. Plotkin J.-L. Lassez, editor, *Computational Logic - Essays in Honor of Alan Robinson*, pages 257–321. The MIT Press, 1991.
12. K. Kunen. Negation in logic programming. *Journal of Logic Programming*, 4:289–308, 1987.
13. J.-L. Lassez, M. J. Maher, and K. Marriott. Unification revisited. In J. Minker, editor, *Foundations of deductive databases and logic programming*, pages 587–625, Los Altos, CA, 1988. Morgan Kaufmann.
14. J.-L. Lassez and K. Marriot. Explicit representation of terms dened by counter examples. *Journal of Automated Reasoning*, 3(13):301–318, 1983.
15. M. J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Proc. of the 3rd IEEE Symp. on Logic in Computer Science*, pages 348–357, 1988.
16. A. I. Malcev. Axiomatizable classes of locally free algebras. In B. F. Wells, editor, *The Metamathematics of Algebraic Systems (Collected Papers: 1936-1967)*, volume 66, chapter 23, pages 262–281. North-Holland, 1971.
17. P. Nickolas. The representation of anwers to logical queries. In *Proceedings of the 11th Australian Computer Science Conference*, pages 246–255, 1988.
18. J.C. Shepherdson. Language and equality theory in logic programming. Technical Report No. PM-91-02, University of Bristol, 1991.
19. M. Tajine. Negation elimination from syntactic equational formula. In C Kirchner, editor, *Proceedings of the 5th International Conference on Rewriting Techniques and Applications*, volume 690 of *Lecture Notes in Computer Science*, pages 316–327, Montreal, Canada, 1993. Springer-Verlag.
20. Dao Thi-Bich-Hanh. Rsolution de contraintes du premier ordre dans la thorie des arbres finis ou infinis. In *Neuvieme Journes Francophones de Programmation Logique et Programmation par Contraintes JFPLC'2000, Marseille 2000*, pages 225–240. Hermes Science Publications, 2000.
21. S. Vorobyov. Theory of finite trees revisited: Aplication of model-theoretic algebra. Technical Report CRIN-94-R-135, Centre de Recherche en Informatique de Nancy, October 1997.