Equational Constraint Solving via a restricted form of universal quantification

Javier Álvez and Paqui Lucio[⋆]

Dpto. de Lenguajes y Sistemas Informáticos, UPV-EHU, Spain.
{jibalgij,jiplucap}@si.ehu.es

Abstract. In this paper, we present a syntactic method for solving first-order equational constraints over term algebras. The presented method exploits a novel notion of *quasi-solved* form that we call *answer*. By allowing a restricted form of universal quantification, *answers* provide a more compact way to represent solutions than the purely existential solved forms found in the literature. *Answers* have been carefully designed to make satisfiability test feasible and also to allow for boolean operations, while maintaining expressiveness and user-friendliness. We present detailed algorithms for (1) satisfiability checking and for performing the boolean operations of (2) negation of one *answer* and (3) conjunction of n *answers*. Based on these three basic operations, our solver turns any equational constraint into a disjunction of *answers*. We have implemented a prototype that is available on the web.

Keywords: equality, constraint satisfaction, solver, term algebra, *answer*.

1 Introduction

An *equational constraint* is an arbitrary first-order formula built over a signature Σ of function symbols and equality as unique predicate symbol. Equational constraints are interpreted over term algebras. An *equational solving method* takes as input an equational constraint and produces the set of all its solutions or, more precisely, some particular representation of it. Syntactic methods are *rewriting processes* that transform the input constraint into an equivalent disjunction of constraints, in the so-called *solved form*, which represents its solutions. In particular, those solutions serve to decide whether the input constraint is satisfiable or not.

On one hand, equational constraint solving is an very important tool in many areas of automated deduction. The integration of efficient equational solvers in theorem provers has been a challenging problem, important for many practical applications. Equational constraints can be used for restricting the set of ground

[⋆] Corresponding author: Javier Álvez, Dpto. de L.S.I., Facultad de Informática, Paseo Manuel de Lardizabal, 1, 20080-San Sebastián, Spain.

instances in order to define more eficient resolution mechanism (cf. [6]). In automated model building, equational constraints play a crucial role for model representation (e.g. [5, 11]). On the other hand, equational constraint solving may be applied for several purposes in the areas of functional/logic programming and databases. Therein, many problems related to semantics and implementation issues can be reduce to equational constraint solving problems. Some well known examples include the problems of answering negative goals, decidindg whether a case-definition is complete, evaluating a boolean conjunctive query on a relational database, etc. Besides, equational constraint solving have been found useful in other areas such as formal verification tools, computational linguistic, machine learning, program transformation, etc.

It is well known that the *free equality theory*[1], originally introduced by Malcev in [15], is non-elementary (see [10, 21]). Besides, the inherent complexity of the satisfiability problem of equational problems (i.e. where the quantifier prefix is of the form $\forall^*\exists^*$) for finite signature is studied in [19]. The most well known algorithms for equational solving [9, 14, 15] and later extensions to richer theories (see [20]) are based on quantifier elimination with solved forms that combine equations and disequations. Negation should be allowed since, for example, the constraint $\forall v(\ x \neq f(v, v)\ )$ cannot be finitely represented without negation (disequations). As opposed to negation, universal quantification can be dropped from any equational formula by the well-known *quantifier elimination technique*. As a consequence, most solved form notions (see [8] for a survey) are boolean combinations of certain kind of existential formulas whose satisfiability test is trivial even in the case of finite signature. However, in exchange for the simplicity of the test, the solver must remove all universal quantifiers. This often requires application of the so-called[2] *Explosion Rule* ([9]) that we recall in Fig. 1, that implies substitution of a formula by a disjunction of as many formulas as there are function symbols in the finite signature Σ. A more compact rep-

$$\text{(Exp)}\quad \forall\overline{y}(\ \varphi\ ) \longmapsto \bigvee_{f\in\Sigma} \exists\overline{z}\forall\overline{y}(\ \varphi \wedge w = f(\overline{z})\ )$$

if there exists an equation $x = t$ or a disequation $x \neq t$ such that some $y_i \in \overline{y}$ occurs in t, $\overline{z}$ are fresh and Σ is finite

Fig. 1. The Explosion Rule (Exp)

resentation of solutions reduces the blow up of the number of disjuncts along the quantifier elimination process, which in turn improves the method. At the same time, the basic operations for managing this more expressive notion must not be expensive. We propose a notion of *quasi-solved form*, called *answer*, that allows a restricted form of universal quantification, which is enough to avoid the

[1] Also called the theory of term algebra and Clark's equational theory.
[2] That is, the *Weak Domain Closure Axiom* in the nomenclature of [14].

above rule (Exp) and offers a more compact representation of solutions. *Answers* have been carefully designed to make satisfiability test feasible and also to allow for boolean operations (of negation and conjunction), while retaining expressiveness and user-friendliness. The idea of gaining efficiency via restricted forms of universal quantification has been already proposed in [18] and in [16].

A very preliminary version of this work was presented as [1]. We have implemented (in Prolog) a prototype of the general constraint solver. It is available at ˜http://www.sc.ehu.es/jiwlucap/equality_constraints.html.

Outline of the paper. In the next section, we recall some useful definitions and denotational conventions. Section 3 is devoted to the details of the notion of *answer* and some examples. In Section 4, we introduce the *answer* satisfiability test with some illustrative examples. In Section 5, we show how the two other basic operations on *answers* —conjunction and negation— can be efficiently performed. Besides, we make use of these basic operations (together with the quantifier elimination technique) to provide a solving method for general equational constraints. We give a summarizing example in Section 6. Finally, we present some concluding remarks and briefly discuss some related work.

2 Definitions and Notation

Let us fix a denumerable set of variables X. Given a (finite or infinite) signature Σ, a Σ-term is a variable from X, or a constant, or a function symbol of arity n applied to n terms. A term is *ground* if it contains no variable symbols. $\mathcal{T}(\Sigma)$ stands for the algebra of all ground Σ-terms or Herbrand universe, whereas $\mathcal{T}(\Sigma, X)$ is used to denote the set of all Σ-terms. We denote by $Var(t)$ the set of all variables occurring in t and $t(\overline{v})$ denotes that $Var(t) \subseteq \overline{v}$. A term is *linear* if it contains no variable repetitions. A bar is used to denote tuples of objects. Subscripts are used to denote the components of a tuple and superscripts are used to enumerate tuples. For example, x_j denotes a component of $\overline{x}$, whereas $\overline{x}^1, \ldots, \overline{x}^j, \ldots, \overline{x}^m$ is a tuple enumeration and x_i^j is a component of the tuple $\overline{x}^j$. Concatenation of tuples is denoted by the infix operator $\cdot$, i.e. $\overline{x} \cdot \overline{y}$ represents the concatenation of $\overline{x}$ and $\overline{y}$. When convenient, we treat a tuple as the set of its components. 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 is also written as $\neg(t_1 = t_2)$). By a *collapsing* equation (or disequation) we mean that at least one of its terms is a variable. We abbreviate collapsing Σ-equation by Σ-CoEq, and Σ-UCD stands for universally quantified collapsing Σ-disequation. We abbreviate $\bigwedge_i t_i = s_i$ by $\overline{t} = \overline{s}$ and $\bigvee_i t_i \neq s_i$ by $\overline{t} \neq \overline{s}$. To avoid confusion, we use the symbol $\equiv$ for the metalanguage equality.

A *Σ-substitution* $\sigma \equiv \{x_1 \leftarrow t_1, \ldots x_n \leftarrow t_n\}$ is a mapping from a finite set of variables $\overline{x}$, called $domain(\sigma)$, into $\mathcal{T}(\Sigma, X)$. It is assumed that σ behaves as the identity for the variables outside $domain(\sigma)$. A substitution σ is called a *Σ-assignment* if $\sigma(x_i) \in \mathcal{T}(\Sigma)$ for all $x_i \in domain(\sigma)$. We intentionally confuse the above substitution σ with the conjunction of equations $\bigwedge_i x_i = t_i$. The (possibly ground) term $\sigma(t)$ (also denoted $t\sigma$) is called an *(ground) instance* of the term

t. The *most general unifier* of a set of terms $\{s_1, \ldots, s_m\}$, denoted $mgu(\overline{s})$, is an idempotent substitution σ such that $\sigma(s_i) \equiv \sigma(s_j)$ for all $1 \leq i, j \leq m$ 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 $1 \leq i \leq n$. The *most general common instance* of two terms t and s, denoted by $mgi(t, s)$, is the term whose set of ground instances is the intersection of both sets of ground instances (for t and s), and it can be computed using a unification algorithm.

An *equational constraint* is a first-order Σ-formula built over Σ-equations (as atoms) using the classical connectives and quantifiers. Atoms include the logical constants True and False. Equational constraints are interpreted in the term algebra $\mathcal{T}(\Sigma)$. A Σ-assignment σ satisfies a Σ-equation $t_1 = t_2$ iff $t_1\sigma \equiv t_2\sigma$. The logical constants, connectives and quantifiers are interpreted as usual. A *solution* of an equational constraint is a Σ-assignment that satisfies the constraint. Constraint equivalence means the coincidence of the set of solutions. We make no distinction between a set of constraints $\{\varphi_1, \ldots, \varphi_k\}$ and the conjunction $\varphi_1 \wedge \ldots \wedge \varphi_k$. We abbreviate $\forall x_1 \ldots \forall x_n$ (resp. $\exists x_1 \ldots \exists x_n$) by $\forall \overline{x}$ (resp. $\exists \overline{x}$).

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.

Definition 1. *Let $\overline{x}$ be a k-tuple of pairwise distinct variables. A Σ-*answer *for $\overline{x}$ is either a logical constant (*True *or* 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 Σ-*CoEq*s and Σ-*UCD*s of the form:*

$$x_1 = t_1 \wedge \ldots \wedge x_k = t_k \ \wedge \ \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m_i} \forall \overline{v}(\ w_i \neq s_{ij}(\overline{w}, \overline{v})\ ) \qquad (1)$$

such that the n-tuple $\overline{w} \equiv Var(t_1, \ldots, t_k)$ is disjoint from $\overline{x}$, every term $s_{ij}(\overline{w}, \overline{v})$ neither contains the variable w_i nor is a single universal variable in $\overline{v}$, and $n, m_1, \ldots, m_n \geq 0$. □

Remark 1. We abbreviate the equational part of (1) by $\overline{x} = \overline{t}(\overline{w})$. Any equation $x_j = w_i$ such that w_i does not occur in the rest of the *answer* can be left out. The scope of each universal quantifier is restricted to one single disequation, although a universal variable can occur repeatedly in a disequation. □

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

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

$$\exists w_1 \exists w_2(\ x = f(w_1, w_2) \wedge \forall v_1(\ w_1 \neq (v_1)\ ) \wedge \forall v_2(\ w_2 \neq f(w_1, v_2)\ )\ ). \qquad (2)$$

By application of the rule (Exp) for eliminating v_1 and v_2, the *answer* (2) is equivalent to the disjunction of the following six existential constraints:

1. $x = f(a, a)$
2. $\exists z_1(\ x = f(a, g(z_1))\ )$
3. $\exists z_1 \exists z_2(\ x = f(a, f(z_1, z_2)) \wedge z_1 \neq a\ )$
4. $\exists z_1 \exists z_2(\ x = f(f(z_1, z_2), a)\ )$
5. $\exists z_1 \exists z_2 \exists z_3(\ x = f(f(z_1, z_2), g(z_3))\ )$
6. $\exists z_1 \exists z_2 \exists z_3 \exists z_4(\ x = f(f(z_1, z_2), f(z_3, z_4)) \wedge z_3 \neq f(z_1, z_2)\ )$ □

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

$$\begin{aligned}\exists w_1 \exists w_2 \forall y_1 \forall y_2(\ & f(f(w_1, a), f(w_2, x_2)) \neq f(f(y_1, a), f(y_2, y_2)) \wedge \\ & f(g(y_2), x_1) \neq f(x_2, f(y_1, y_1))\ )\end{aligned}$$

is equivalent to the disjunction of the following two *answers*:[3]

$$\exists w_2(\ x_2 = w_2 \wedge \forall v(\ w_2 \neq g(v)\ )\ ) \tag{3}$$
$$\exists w_1(\ x_1 = w_1 \wedge \forall v(\ w_1 \neq f(v, v)\ )\ ) \tag{4}$$

It is easy to see that the equational part of any *answer* is always satisfiable. In fact, it is an idempotent substitution. Besides, with infinitely many function symbols, one can always find an assignment that satisfies a given finite conjunction of UCDs. Therefore, *answers* are always satisfiable for infinite signatures. On the contrary, if Σ is finite, the disequational part of an *answer* is not necessarily satisfiable in $\mathcal{T}(\Sigma)$. As a consequence, we say that an *answer* is a *quasi-solved form.*

4 The Satisfiability Test

In this section, we introduce an algorithm for deciding *answer* satisfiability w.r.t. a finite signature. Notice that, for finite Σ, the set $\mathcal{T}(\Sigma)$ can be finite or infinite. The test works for both. We also give some examples of satisfiability and discuss about efficiency. In the case of finite signature Σ, as explained above, an *answer* is satisfiable iff its disequational part is. Hence, we shall concentrate in the disequational part of the tested *answer*. A Σ-*answer* $\exists \overline{w}(\ a(\overline{x}, \overline{w})\ )$ is satisfiable in $\mathcal{T}(\Sigma)$ iff there is at least one Σ-assignment with domain $\overline{w}$ that satisfies the conjunction of UCDs inside $a(\overline{x}, \overline{w})$.

Example 3. Each of the two following conjunctions of Σ-UCDs is (individually) unsatisfiable in $\mathcal{T}(\Sigma)$ (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. $w \neq a \wedge w \neq g(a) \wedge \forall v(\ w \neq g(g(v))\ ) \wedge$
   $\forall v_1 \forall v_2(\ w \neq g(f(v_1, v_2))\ ) \wedge \forall v_1 \forall v_2(\ w \neq f(v_1, v_2)\ )$ □

[3] As said in Remark 1, we have left out $x_1 = w_1$ in (3) and $x_2 = w_2$ in (4).

The satisfiability of a conjunction of UCDs on a variable w_i could be tested using the algorithm introduced in [13] for deciding whether an implicit representation of terms can be explicitely (finitely) represented. In [13], an implicit representation is an expresion $t/t_1 \vee \ldots \vee t_n$ that represents those ground instances of t that are not instances of any t_i for $1 \leq i \leq n$. Actually, we could transform any set

$$\mathsf{S} \equiv \{\forall \overline{v}(\ w_i \neq s_{ij}(\overline{w}, \overline{v})\ ) \mid 1 \leq i \leq n,\ 1 \leq j \leq m_i\}$$

of UCDs into an equivalent[4] set of inequalities on the involved tuple $\overline{w}$, using a fresh function symbol $\underline{\mathsf{c}}$ as tuple constructor and fresh variables $\overline{u}$, as follows:

$$\begin{aligned}\mathsf{Tup}(\mathsf{S}) \equiv \{\ & \forall \overline{v} \forall \overline{u}(\ \underline{\mathsf{c}}(\overline{w}) \neq \underline{\mathsf{c}}(\overline{w}\sigma_{ij})\ ) \mid 1 \leq i \leq n,\ 1 \leq j \leq m_i, \\ & \sigma_{ij} \equiv \theta_{ij} \cup \{w_i \leftarrow s_{ij}\theta_{ij}\} \text{ where } \theta_{ij} \equiv \{w_k \leftarrow u_k | 1 \leq k \neq i \leq m_j\}\}.\end{aligned} \tag{5}$$

Then, by treating the tuple constructor $\underline{\mathsf{c}}$ as just another function symbol (except for the explosion rule), the algorithm *uncover* of [13] could be invoked, as $\mathit{uncover}(\underline{\mathsf{c}}(\overline{w}), \underline{\mathsf{c}}(\overline{w})\sigma_{11}, \ldots, \underline{\mathsf{c}}(\overline{w})\sigma_{nm_n})$, in order to decide if there exists an explicit representation of the implicit representation $\underline{\mathsf{c}}(\overline{w}) \backslash \underline{\mathsf{c}}(\overline{w})\sigma_{11} \vee \ldots \vee \underline{\mathsf{c}}(\overline{w})\sigma_{nm_n}$. The algorithm *uncover* refines such implicit representations to explicit ones (if it is possible), by applying the explosion rule only to linear[5] terms. For example, being the signature $\{a/0, b/0, f/2\}$, the implicit representation

$$\underline{\mathsf{c}}(v_1, a) \setminus \underline{\mathsf{c}}(a, a) \vee \underline{\mathsf{c}}(b, v_2)$$

can be explicitly represented by $\{\underline{\mathsf{c}}(f(u_1, u_2), a)\}$, whereas $\underline{\mathsf{c}}(v_1, v_2) \setminus \underline{\mathsf{c}}(v_3, v_3)$ cannot be explicitly represented.

Our *answer* satisfiability test exploits the same idea but, for efficiency, it does not exhaustively obtain the set of all possible elements represented by the tuple of variables $\underline{\mathsf{c}}(\overline{w})$, as the call $\mathit{uncover}(\underline{\mathsf{c}}(\overline{w}), \underline{\mathsf{c}}(\overline{w})\sigma_{11}, \ldots, \underline{\mathsf{c}}(\overline{w})\sigma_{nm_n})$ does. Instead, it works by value elimination in two steps. Each step takes into account a different (syntactic) subclass of UCDs. Moreover, we use a slightly simplified version of *uncover* (see Figure 2) since term linearity is an invariant condition in our *uncover* runs. Figure 3 outlines our *answer* satisfiability test for a finite signature Σ. We denote by $Ext(w_i)$ —namely, the extension of w_i— the collection of all ground terms that w_i could take as value. As in [13], we use fresh variables for finitely representing infinite variable extensions. For instance, $\{v\}$ represents the whole $\mathcal{T}(\Sigma)$'s universe (for any Σ) and $\{f(v_1, a), f(b, v_2)\}$ an infinite subset when Σ contains $\{a, b, f\}$. The basic idea is that a UCD disallows some values in $Ext(w_i)$. The first step treats the UCDs $\forall \overline{v}(\ w_i \neq s_{ij}(\overline{v})\ )$ without existential variables (that is, $Var(s_{ij}) \cap \overline{w} \equiv \emptyset$) and without repetitions in their universal variables $\overline{v}$. Roughly speaking, this is the only subclass of UCDs that can transform an infinite variable extension —described by a collection of linear terms— into a finite one. After the first iteration in Step 1 (see Fig. 3), the extension of each w_i can either be empty, or non-empty finite, or infinite. Notice

[4] This well known equivalence corresponds to the rule U_2 in [9].

[5] In [13], linear terms are called unrestricted terms.

```
$partition(t,\theta)$ is    // $\overline{u}$ stands for a fresh tuple of variables of the adequate size
    if $\theta$ is a renaming then return $\emptyset$
    else select $\{z \leftarrow f(s_1,\dots s_h)\} \in \theta$
        let $\sigma' \equiv \{z \leftarrow f(\overline{u})\}$ and $\sigma_i \equiv \{u_i \leftarrow s_i\}$ for $1 \leq i \leq h$
        return $\bigcup_{g\in\Sigma, g\neq f} t\{z \leftarrow g(\overline{u})\} \cup partition(t\sigma', \theta \setminus \sigma \cup \{\sigma_i | 1 \leq i \leq h\})$

$uncover(t, t\theta_1, \dots, t\theta_n)$ is
    let $\{s_1,\dots,s_k\} \equiv partition(t,\theta_1)$
    let $\sigma_{ij} \equiv mgi(s_i, t\theta_j)$ for each $(i,j) \in \{1,\dots,k\} \times \{2,\dots,n\}$
    return $\bigcup_{r=1}^{k} uncover(s_r, s_r\sigma_{r2}, \dots, s_r\sigma_{rn})$
```

Fig. 2. Simplified Version of *uncover* ([13])

that $Ext(w_i)$ represents an infinite set of terms if and only if it contains (at least) one non-ground term. Then, the input *answer* is unsatisfiable if some $Ext(w_i) \equiv \emptyset$. On the contrary, the *answer* is satisfiable if every $Ext(w_i)$ is infinite. This first step is very often enough to decide the satisfiability of the input *answer*.

Example 4. The above *answers* (2), (3) and (4) are decided to be satisfiable at the first step since both $Ext(w_1)$ and $Ext(w_2)$ remain infinite. For (2) and (3), the test takes into account the first UCD (the only one in (3)). For (4), no UCD satisfies the condition, hence the test decides at once. □

Example 5. Both constraints in Example 3 are unsatisfiable. This is decided in the first step since $Ext(w)$ becomes empty. □

At the end of the first step, if no $Ext(w_i)$ is empty and at least one $Ext(w_i)$ is finite, the test requires a second step. The second step looks whether there exists (at least) one assignment σ with domain $\overline{w}^{fin} \equiv \{w_i | Ext(w_i) \text{ is finite}\}$ that satisfies only the subclass FinUCD of the remaining UCDs such that all their existential variables are contained in $\overline{w}^{fin}$. We denote by $Ext(\overline{w}^{fin})$ the cartesian product of all $Ext(w_i)$ such that $w_i \in \overline{w}^{fin}$.

Example 6. Let $\Sigma \equiv \{a/0, g/1, f/2\}$. Suppose that S is a set of UCDs formed by $\forall v_1(\ w_1 \neq g(v_1)\ )$, $\forall v_1 \forall v_2(\ w_1 \neq f(v_1, v_2)\ )$ and a large finite set S_0 of UCDs of the form $\forall v(\ w_i \neq f(w_j, v)\ )$ where $i, j > 1$ and $i \neq j$. At the first step, only the two first UCDs are considered to yield $Ext(w_1) \equiv \{a\}$, whereas $Ext(w_i)$ is infinite for all $i > 1$. At the second step, FinUCD becomes empty. Hence, S is satisfiable. Notice that the exhaustive computation of *uncover* over the whole tuple of variables could be much more expensive depending on the size of S_0. □

Theorem 1. *The satisfiability test of Figure 3 terminates for any input* (S, Σ)*. Besides, it returns "satisfiable" if there exists a* Σ*-assignment that satisfies the set* S*. Otherwise, it returns "unsatisfiable".*

Proof. It is based on the results of [13]. See the appendix. □

```
// input: $\mathsf{S} \equiv \{\forall\overline{v}(\ w_i \neq s_{ij}(\overline{w},\overline{v})\ ) \mid 1 \leq i \leq n,\ 1 \leq j \leq m_i\}$
//        $\Sigma \equiv \{f_1\backslash a_1,\ldots,f_k\backslash a_k\}$   (finite signature)

step 1: for $i \in \{1,\ldots,n\}$ do
            let $\{t_1,\ldots t_{k_i}\} \equiv \{s_{ij} \mid 1 \leq j \leq m_i, Var(s_{ij}) \cap \overline{w} \equiv \emptyset$ and each
                                        $v_h \in \overline{v}$ occurs at most once in $s_{ij}\}$
            if $k_i = 0$ then $Ext(w_i) := \{u\}$ for fresh $u$
                       else $Ext(w_i) := uncover(w_i, t_1, \ldots, t_{k_i})$
            when some $Ext(w_i) \equiv \emptyset$ exit with *unsatisfiable*
        let $\overline{w}^{fin}$ be the tuple of all $w_i \in \overline{w}$ such that $Ext(w_i)$ is finite
        if $\overline{w}^{fin} \equiv \emptyset$ then exit with *satisfiable*
                   else go to step 2

step 2: let $\mathsf{FinUCD} \equiv \{\forall\overline{v}(\ w_i \neq s_{ij}(\overline{w},\overline{v})\ ) \in \mathsf{S} \mid w_i \in \overline{w}^{fin}, Var(s_{ij}) \setminus \overline{v} \subseteq \overline{w}^{fin}\}$
        if $\mathsf{FinUCD} \equiv \emptyset$ then exit with *satisfiable*
        else let $\{\sigma_1,\ldots,\sigma_m\}$ be the set of substitutions such that
                 $\mathsf{Tup}(\mathsf{FinUCD}) \equiv \{\forall\overline{v}\forall\overline{u}(\ \overline{w} \neq \overline{w}\sigma_i\ ) \mid 1 \leq i \leq m\}$        (see (6))
            $C := \bigcup_{\overline{t} \in Ext(\overline{w}^{fin})} uncover(\overline{t}, \overline{t}\sigma_1, \ldots, \overline{t}\sigma_m)$
            if $C \not\equiv \emptyset$ then exit with *satisfiable*
                       else exit with *unsatisfiable*
```

Fig. 3. *Answer* Satisfiability Test

The introduced satisfiability test has a poor worst case performance. Actually, *answer* satisfiability is an NP-complete problem (see [19]). However, the test performs efficiently in practice because of several structural reasons that can be summed up as follows. In general, *answers* having expensive computations in both steps are unlikely. If the input *answer* contains a lot of UCDs to be treated in the first step, the extension of some variable usually becomes empty and the test stops. However, where few UCDs are treated at the first step, it is usual that most extensions remain infinite and, therefore, the second step becomes unnecessary or very cheap.[6] On the contrary, the worst case occurs when every w_i has a large finite extension, but every possible assignment violates some UCD. The combination of both properties requires a lot of UCDs to be expressed.[7]

5 Operations on *Answers* and Equational Solving

In this section, we present a method for transforming any equational constraint into a disjunction of *answers*. This solving method uses, besides the satisfiability test, two boolean operations on *answers* —conjunction and negation— which will also be presented. For general equational solving, we use the classical quantifier elimination technique. However, we keep the matrix as a disjunction of

[6] Remember that the second step only checks the UCDs such that all their existential variables have a finite extension (according to the first step).

[7] Since the first-order language of the free equality cannot express very "deep" properties of terms.

satisfiable *answers* on the prefix variables, instead of a quantifier-free CNF (or DNF) formula. That is, at every step we have a formula of the form:

$$Q_1\overline{u}^1 \dots Q_m\overline{u}^m(\bigvee_{j=1}^{k} \exists\overline{w}(\ a_j(\overline{u}^1 \cdot \dots \cdot \overline{u}^m, \overline{w})\ )\ )$$

where each $Q_i \in \{\forall, \exists\}$ and each $a_j(\overline{u}^1 \cdot \dots \cdot \overline{u}^m, \overline{w})$ is a satisfiable *answer* on $\overline{u}^1 \cdot \dots \cdot \overline{u}^m$. Then, if the last block Q_m is $\exists$, it is easily eliminated by erasing from each a_j all the equations on $\overline{u}^m$. Then, we also remove every UCD containing at least one w_k which does not occur in the equational part.[8] It is worthwhile to notice that the satisfiability of each $a_j(\overline{u}^1 \cdot \dots \cdot \overline{u}^m, \overline{w})$ guarantees that both the eliminated and the remaining part of the treated *answer* are satisfiable. In fact, the *answer* is equivalent to $(\exists\overline{u}^m\varphi_1) \wedge \varphi_2$, where $(\exists\overline{u}^m\varphi_1)$ is the eliminated part and φ_2 the remaining one. If $Q_m \equiv \forall$, double negation is applied:

$$Q_1\overline{u}^1 \dots Q_{m-1}\overline{u}^{m-1}\neg\exists\overline{u}^m\neg(\bigvee_{j=1}^{k} \underbrace{\exists\overline{w}(\ a_j(\overline{u}^1 \cdot \dots \cdot \overline{u}^m, \overline{w})\ )}_{\psi_j}\ ). \tag{6}$$

Suppose a procedure P exists that transforms the negation of a disjunction of *answers* (on some variables) into a disjunction of *answers* (on the same variables). Then, using P, the inner formula $\neg\bigvee_{j=1}^{k}\psi_j$ is transformed into a disjunction of *answers*. After that, $\exists\overline{u}^m$ is easily eliminated as above. Finally, the procedure P is again applied and the original innermost block $\forall\overline{u}^m$ is already eliminated. We implement the procedure P as follows, using two boolean operations on *answers*:

$$\neg\bigvee_{j=1}^{k}\psi_j \longmapsto \bigwedge_{j=1}^{k}\neg\psi_j \longmapsto^{\mathbf{(1)}} \bigwedge_{j=1}^{k}\bigvee_{r=1}^{m_j}\varphi_{jr} \longmapsto \bigvee_{r=1}^{m}\bigwedge_{j=1}^{k_r}\beta_{jr} \longmapsto^{\mathbf{(2)}} \bigvee_{r=1}^{m}\bigvee_{h=1}^{n_r}\gamma_{hr}$$

(1) The negation of an *answer* ψ_j yields a disjunction $\bigvee_{r=1}^{m_j}\varphi_{jr}$ of *answers*.

(2) The conjunction $\bigwedge_{j=1}^{k_r}\beta_{jr}$ of *answers* yields a disjunction $\bigvee_{h=1}^{n_r}\gamma_{hr}$ of *answers*.

Moreover, both boolean operations preserve the variables for which source and target *answers* are obtained. In the next two subsections, we give the successive steps that constitute each transformation. It is easy to see that each step preserves equivalence. For that, we use the auxiliary transformation rule (UD) of Fig. 4.

Proposition 1. *The transformation rule (UD) of Figure 4 is correct.*

Proof. See the appendix. □

[8] It has just disappeared with the previously eliminated equations.

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

Fig. 4. Auxiliary Transformation Rule (UD)

5.1 Negation of one *answer*

Now, we show how to transform the negation of an *answer* for $\overline{x}$ into an equivalent disjunction of *answers* for $\overline{x}$. First, we apply the transformation rule (UD):

$$\neg\exists\overline{w}[\ \overline{x}=\overline{t}(\overline{w})\wedge\bigwedge_{i=1}^{n}\bigwedge_{j=1}^{m_i}\forall\overline{v}(\ w_i\neq s_{ij}(\overline{w},\overline{v})\ )\ ]\longmapsto_{(UD)}$$
$$\neg\exists\overline{w}[\ \overline{x}=\overline{t}(\overline{w})\ ]\vee\exists\overline{w}[\ \overline{x}=\overline{t}(\overline{w})\wedge\bigvee_{i=1}^{n}\bigvee_{j=1}^{m_i}\exists\overline{v}(\ w_i=s_{ij}(\overline{w},\overline{v})\ )\ ].$$

By the rule (UD), the first disjunct $\neg\exists\overline{w}(\ \overline{x}=\overline{t}(\overline{w})\ )$ is transformed into:

$$\begin{aligned}&\forall\overline{w}^1[\ x_1\neq t_1(\overline{w}^1)\ ]\vee\exists\overline{w}^1[\ x_1=t_1(\overline{w}^1)\wedge\forall\overline{w}^2(\ x_2\neq t_2(\overline{w}^1\cdot\overline{w}^2)\ )\ ]\vee\ \dots\ \vee\\&\exists\overline{w}^1\dots\exists\overline{w}^{n-1}[\ x_1=t_1(\overline{w}^1\cdot\dots\cdot\overline{w}^{n-1})\wedge\ \dots\ \wedge x_{n-1}=t_{n-1}(\overline{w}^1\cdot\dots\cdot\overline{w}^{n-1})\\&\qquad\wedge\forall\overline{w}^n(\ x_n\neq t_n(\overline{w}^1\cdot\dots\cdot\overline{w}^n)\ )\ ]\end{aligned}\qquad(7)$$

Then, we replace the variables x_i in the disequations by new variables w'_i, adding the corresponding equation $x_i=w'_i$. The result is already a disjunction of *answers* for $\overline{x}$. For the second disjunct, we first lift the internal disjunctions:

$$\bigvee_{i=1}^{n}\bigvee_{j=1}^{m_i}\exists\overline{w}[\ \overline{x}=\overline{t}(\overline{w})\wedge\exists\overline{v}(\ w_i=s_{ij}(\overline{w},\overline{v})\ )\ ]$$

and then substitute each $s_{ij}(\overline{w},\overline{v})$ for w_i in $t(\overline{w})$:

$$\bigvee_{i=1}^{n}\bigvee_{j=1}^{m_i}\exists\overline{w}\exists\overline{v}(\ \overline{x}=\overline{t}(\overline{w})\{w_i\leftarrow s_{ij}(\overline{w},\overline{v})\}\ ).$$

5.2 Conjunction of *k answers*

Using unification, we transform a conjunction of k *answers* for $\overline{x}$:

$$\bigwedge_{i=1}^{k}\exists\overline{w}^i[\ \overline{x}=\overline{t}^i(\overline{w}^i)\wedge\bigwedge_{h=1}^{n}\bigwedge_{j=1}^{m_h}\forall\overline{v}(\ w_h^i\neq s_{hj}^i(\overline{w}^i,\overline{v})\ )\ ]$$

into an equivalent disjunction of *answers* for $\overline{x}$. If the $mgu(\overline{t}_1(\overline{w}^1),\dots,\overline{t}_k(\overline{w}^k))$ does not exist, the disjunction is equivalent to False. Otherwise, we get a substitution σ that is used for joining the equational parts as follows:

$$\exists\overline{w}^1\dots\exists\overline{w}^k[\ \overline{x}=\sigma(\overline{t}^1(\overline{w}^1))\wedge\bigwedge_{h=1}^{n}\bigwedge_{j=1}^{m_h}\forall\overline{v}(\ \sigma(w_h^i)\neq\sigma(s_{hj}^i(\overline{w}^i,\overline{v}))\ )\ ].$$

Now, letting $\overline{w} \equiv \overline{w}^1 \cdot \ldots \cdot \overline{w}^k$ we have a constraint of the form:

$$\exists \overline{w}[\ \overline{x} = t'(\overline{w}) \wedge \bigwedge_{h=1}^{n} \bigwedge_{j=1}^{m_h} \neg \exists \overline{v}(\ t_h(\overline{w}) = r_{hj}(\overline{w}, \overline{v})\ )\ ].$$

Let $\sigma_{hj} \equiv mgu(t_h(\overline{w}), r_{hj}(\overline{w}, \overline{v}))$. If σ_{hj} does not exist, $\neg \exists \overline{v}(\ t_h(\overline{w}) = r_{hj}(\overline{w}, \overline{v})\ )$ is equivalent to True. Otherwise, since σ_{hj} is an idempotent substitution, each answer $\neg \exists \overline{v}(\ \sigma_{hj}\ )$ can be transformed into a disjunction $\bigvee_{k=1}^{n_{hj}} \exists \overline{z}(\ a_k(\overline{w}, \overline{z})\ )$ of *answers* for $\overline{w}$ by (UD) as we show in Subsect. 5.1 (see (7)). Hence, the constraint has the form:

$$\exists \overline{w}[\ \overline{x} = t'(\overline{w}) \wedge \bigwedge_{h=1}^{n} \bigwedge_{j=1}^{m_h} \bigvee_{k=1}^{n_{hj}} \exists \overline{z}(\ a_k(\overline{w}, \overline{z})\ )\ ]. \qquad (8)$$

Finally, we apply distribution and (recursively) conjunction of *answers* for $\overline{w}$:

$$\exists \overline{w}[\ \overline{x} = t'(\overline{w}) \wedge \bigvee_{r=1}^{m} \exists \overline{z}(\ b_r(\overline{w}, \overline{z})\ )\ ].$$

Then, we lift the internal disjunction and substitute the equational part of each *answer* $b_r(\overline{w}, \overline{z})$ in $t'(\overline{w})$.

Notice that, the only blow-up (in the number of *answers*) could be produced in the transformation of (8), by the distribution and conjuntion of *answers*. However, in practice, the blow-up is often non-significant. This is due to the fact that the above $\bigvee_{k=1}^{n_{hj}} \exists \overline{z}(a_k(\overline{w}, \overline{z}))$ are obtained by using the rule (UD). Since the rule (UD) yields mutually excluding constraints, many internal conjunctions of *answers* for $\overline{w}$ are reduced to False at once.

6 A Complete Example

In this section, we demonstrate the application of our solving method to the following equational constraint on x_1, x_2, x_3 (free variables):

$$\begin{aligned} \forall y_1 \exists w_1 \forall y_2 (\ & f(x_1, g(y_2)) \neq f(f(w_1, x_2), a) \wedge w_1 \neq f(y_1, y_1) \wedge \\ & \exists w_2 \forall y_3 (\ f(x_2, a) \neq f(g(y_3), w_1)\ ) \wedge \\ & \forall y_4 \forall y_5 (\ f(x_1, x_2) \neq f(g(x_3), f(y_4, y_5))\ )\ \ ) \end{aligned}$$

First, after a preliminary treatment, we obtain the following disjunction of *answers* for x_1, x_2, x_3, y_1, w_1 (as matrix) prefixed by $\forall y_1 \exists w_1$:

$$\begin{aligned} \forall y_1 \exists w_1 (\ & \exists \overline{z}(\ x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge y_1 = z_4 \wedge w_1 = z_5 \wedge \\ & \qquad z_5 \neq f(z_4, z_4) \wedge \forall y_4 \forall y_5 (\ z_2 \neq f(y_4, y_5)\ ) \wedge \forall y_2 (\ z_2 \neq g(y_2)\ )\ )\ ) \vee \\ & \exists \overline{z}(\ x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge y_1 = z_4 \wedge w_1 = z_5 \wedge \\ & \qquad z_5 \neq f(z_4, z_4) \wedge z_1 \neq g(z_3) \wedge \forall y_3 (\ z_2 \neq g(y_3)\ )\ )\ ) \vee \\ & \exists \overline{z}(\ x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge y_1 = z_4 \wedge w_1 = z_5 \wedge \\ & \qquad z_5 \neq f(z_4, z_4) \wedge \forall y_4 \forall y_5 (\ z_2 \neq f(y_4, y_5)\ ) \wedge z_5 \neq a\ ) \vee \\ & \exists \overline{z}(\ x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge y_1 = z_4 \wedge w_1 = z_5 \wedge \\ & \qquad z_5 \neq f(z_4, z_4) \wedge z_1 \neq g(z_3) \wedge z_5 \neq a\ )\ ). \end{aligned}$$

Notice that the prefix is shorter (than the prefix of the prenex-DNF form) because *answers* allow universal quantification.

Now, quantifier elimination is successively applied until the prefix is erased. The innermost block $\exists w_1$ is easily eliminated by removing the CoEq $w_1 = z_5$ and all the UCDs involving z_5. Thus, 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:

$$\begin{aligned}
\neg\exists y_1(\ \ & \neg\exists\overline{z}(\ x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge y_1 = z_4 \wedge \forall y_4 \forall y_5(\ z_2 \neq f(y_4, y_5)\ ) \wedge \\
& \qquad\qquad \forall y_2(\ z_2 \neq g(y_2)\ )\ ) \wedge \\
& \neg\exists\overline{z}(\ x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge y_1 = z_4 \wedge z_1 \neq g(z_3) \wedge \\
& \qquad\qquad \forall y_3(\ z_2 \neq g(y_3)\ )\ ) \wedge \\
& \neg\exists\overline{z}(\ x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge y_1 = z_4 \wedge \forall y_4 \forall y_5(\ z_2 \neq f(y_4, y_5)\ )\ ) \wedge \\
& \neg\exists\overline{z}(\ x_1 = z_1 \wedge x_2 = z_2 \wedge x_3 = z_3 \wedge y_1 = z_4 \wedge z_1 \neq g(z_3)\ )\ ).
\end{aligned}$$

Then, each of the four negated *answers* for the variables x_1, x_2, x_3, y_1 produces a disjunction of *answers* for the same variables, as follows:

$$\begin{aligned}
\neg\exists y_1(\ [\ & \exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5) \vee \\
& \exists\overline{z}(x_1 = z_1 \wedge x_2 = g(z_2) \wedge x_3 = z_3 \wedge y_1 = z_4)\ ] \wedge \\
[\ & \exists\overline{z}(x_1 = g(z_1) \wedge x_2 = z_2 \wedge x_3 = z_1 \wedge y_1 = z_4) \vee \\
& \exists\overline{z}(x_1 = z_1 \wedge x_2 = g(z_2) \wedge x_3 = z_3 \wedge y_1 = z_4)\ ] \wedge \\
[\ & \exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5)\ ] \wedge \\
[\ & \exists\overline{z}(x_1 = g(z_1) \wedge x_2 = z_2 \wedge x_3 = z_1 \wedge y_1 = z_3)\ ]\ )
\end{aligned}$$

By distributing conjunction over disjunction, we obtain the following disjunction of conjunctions of *answers*:

$$\begin{aligned}
\neg\exists y_1(\ [\ & \exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5) \wedge \\
& \exists\overline{z}(x_1 = g(z_1) \wedge x_2 = z_2 \wedge x_3 = z_1 \wedge y_1 = z_4) \wedge \\
& \exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5) \wedge \\
& \exists\overline{z}(x_1 = g(z_1) \wedge x_2 = z_2 \wedge x_3 = z_1 \wedge y_1 = z_3)\ ] \vee \\
[\ & \exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5) \wedge \\
& \exists\overline{z}(x_1 = z_1 \wedge x_2 = g(z_2) \wedge x_3 = z_3 \wedge y_1 = z_4) \wedge \\
& \exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5) \wedge \\
& \exists\overline{z}(x_1 = g(z_1) \wedge x_2 = z_2 \wedge x_3 = z_1 \wedge y_1 = z_3)\ ] \vee \\
[\ & \exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5) \wedge \\
& \exists\overline{z}(x_1 = z_1 \wedge x_2 = g(z_2) \wedge x_3 = z_3 \wedge y_1 = z_4) \wedge \\
& \exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5) \wedge \\
& \exists\overline{z}(x_1 = g(z_1) \wedge x_2 = z_2 \wedge x_3 = z_1 \wedge y_1 = z_3)\ ] \vee \\
[\ & \exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5) \wedge \\
& \exists\overline{z}(x_1 = z_1 \wedge x_2 = g(z_2) \wedge x_3 = z_3 \wedge y_1 = z_4) \wedge
\end{aligned}$$

$$\exists\overline{z}(x_1 = z_1 \wedge x_2 = f(z_2, z_3) \wedge x_3 = z_4 \wedge y_1 = z_5) \wedge$$
$$\exists\overline{z}(x_1 = g(z_1) \wedge x_2 = z_2 \wedge x_3 = z_1 \wedge y_1 = z_3)\ ]\ ).$$

It is easy to see that, in the latter three conjunctions of *answers*, the variable x_2 makes impossible the unification that is required to perform a conjunction of *answers*. Hence, the three conjunctions are transformed to False at once. Whereas the first conjunction yields the following satisfiable *answer* for $\overline{x} \cdot y_1$:

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

Then, the block $\exists y_1$ and the equation $y_1 = z_4$ can be eliminated. Finally, the negation of the resulting *answer* for $\overline{x}$, that is:

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

yields the following disjunction of two *answers* for $\overline{x}$:

$$\exists\overline{w}(\ x_1 = w_4 \wedge x_2 = f(w_2, w_3) \wedge x_3 = w_1 \wedge w_4 \neq g(w_1)\ ) \vee$$
$$\exists\overline{w}(\ x_1 = w_2 \wedge x_2 = w_3 \wedge x_3 = w_1 \wedge \forall v_1 \forall v_2(\ w_3 \neq f(v_1, v_2)\ )\ ).$$

7 Conclusions and Related Work

The notion of *answer* provides a sufficiently compact representation of solutions while retaining user-friendliness and efficient performance of basic operations. In particular, we give detailed algorithms for *answer* satisfiability checking (for finite signature), negation of one *answer* and conjunction of several *answers*. This combination of features makes *answers* a suitable notion of (quasi-)solved form for achieving a good trade-off between time and space efficiency in theorem proving methods for logics with equality. *Answers* are particularly suitable for methods that require some equality constraint notation more expressive than substitutions. We have shown how the quantifier elimination method takes advantage of using *answers* in this sense, given a new method for general equality constraint solving. This method applies to both finite and infinite signatures. The only difference is that satisfiability checking is not needed in the latter case.

Answer is an intermediate notion between purely existential solved forms of, for example, [14, 9] and *substitutions with exceptions* of [4]. *Answers* allow a kind of restricted universal quantification which, besides being more expressive, allows one to confine the role of the explosion rule to the satisfiability test. In this process, since universal quantifiers are not eliminated, we never blow up the tested *answer* via the explosion rule. Explosion is only implicitly used (in the satisfiability test) for computing the indispensable variable extensions. The methods presented in [14] and [9] are both based in quantifier elimination with explicit usage of the explosion rule, although they use two different notions of solved form. The method of [14], for finite signatures, is based on using the explosion rule to perform the conjunction of boolean combinations of basic formulas. For instance, $\exists w(\ x = w\ ) \wedge \neg\exists u(\ x = f(g(f(u, u)), u)\ )$ is solved by explosion of the first conjunct and then by unification. This operation produces a (signature-dependent) number of existential disjuncts. For example, if the signature also contains the constant a, it yields $x = a, \exists w(x = f(a, w)), \exists w(x = f(g(a), w)), \ldots$. Our method yields a (signature-independent) disjunction of *answers*, that is $\exists w(\ x = w \wedge \forall u(\ w \neq f(g(f(u, u)), u)\ )\ )$ for the just above example. Similarly, the proposal of [9], which has been implemented by N. Peltier (see [16, 17]), uses explosion to eliminate all the universal quantifiers. Explosion increases the number of disjuncts in a ratio proportional to the signature. Besides, this blow up interacts with the "CNF-to-DNF" transformation. We cannot avoid the latter blow up, but we benefit from the smaller number of conjuncts. As a consequence, our solutions are always (except for very simple examples) shorter and computationally simpler than the ones given by the system in [17]. Actually, to improve this system, a (very different from ours) restricted form of universal quantification is proposed in [16]. This new solved form allows to replace explicit explosion with the so-called *binary explosion*, which is signature-independent and yields a binary blow up of the formula. Unfortunately, this improvement has not been yet incorporated to [17].

The closest work to our own can be found in [4], where two notions of solved form are provided. They are called *substitutions with exceptions* (SwEs for short) and *constrained substitution*. Both involve universal quantification and require a satisfiability test in the case of finite signature. However, there are significant differences, the most important one being that universal quantification is more restricted in *answers* than in both solved forms of [4]. The following discussion is applicable to both notions of solved form, even though in the sequel we will only mention SwEs. With respect to *answers*, SwEs provide a more compact representation of solutions, but the basic operations for handling them, in particular the satisfiability test, become intricate. More precisely, a SwE is an expression of the form $\sigma_0[\overline{x}, \overline{w}^0] - \{\sigma_1[\overline{x}, \overline{w}^1], \ldots, \sigma_k[\overline{x}, \overline{w}^k]\}$ where $\sigma[\overline{z}, \overline{y}]$ denotes a substitution on domain $\overline{z}$ such that $\overline{y} \equiv Var(\sigma(\overline{z}))$. A SwE of the above form is interpreted as the equality constraint $\exists \overline{w}^0(\ \sigma_0 \wedge \forall \overline{w}^1(\ \neg\sigma_1\ ) \wedge \ldots \wedge \forall \overline{w}^k(\ \neg\sigma_k\ )\ )$. Notice that each $\neg\sigma_i$ is a disjunction of disequations. For example, the following SwE:

$$\begin{aligned}&\{x_1 \leftarrow f(a, w_1), x_2 \leftarrow g(w_1), x_3 \leftarrow f(w_2, w_1)\}\\ &\ -\{\ \{x_1 \leftarrow f(a, y), x_2 \leftarrow f(a, y), x_3 \leftarrow g(y)\},\\ &\qquad \{x_1 \leftarrow f(a, g(y_1)), x_2 \leftarrow g(y_2) \leftarrow x_3, f(y_3, v)\}\ \}\end{aligned}$$

corresponds to the equality constraint:

$$\begin{aligned}\exists w_1 \exists w_2(\ \ & x_1 = f(a, w_1) \wedge x_2 = g(w_1) \wedge x_3 = f(w_2, w_1) \wedge \\ & \forall y(\ x_1 \neq f(a, y) \vee x_2 \neq f(a, y) \vee x_3 \neq g(y)\ ) \wedge \\ & \forall y_1 \forall y_2 \forall y_3 \forall v(\ x_1 \neq f(a, g(y_1)) \vee x_2 \neq g(y_2) \vee x_3 \neq f(y_3, v)\ )\ \ )\end{aligned} \tag{9}$$

In *answers*, universal quantification is restricted to affect one disequation, instead of a disjunction of disequations. Since universal quantification does not distribute over disjunction, this is not a minor difference, especially when testing satisfiability. Actually, [4] introduces a method for solving a *system of equations* *and disequations* with the proviso that a satisfiability test is given. There, it is shown that, for testing satisfiability, it is not enough to check that a substitution is an instance of another. Instead, it is necessary to check whether each instance of the former is an instance of the latter substitution that, in general, requires an infinite number of checks. The solving method that we introduce here provides an easy way for transforming any SwE into a (possibly empty, if unsatisfiable) disjunction of satisfiable *answers*. For example, our constraint solver transforms the above SwE, really the equality constraint (9), into the following *answer*:

$$\exists w_1 \exists w_2 (\ x_3 = f(w_2, w_1) \wedge x_2 = g(w_1) \wedge x_1 = f(a, w_1) \wedge \forall v_1 (\ w_1 \neq g(v_1)\ )\ )$$

Two other notions of solved form that allow for restricted forms of universal quantification were introduced in [18, 19] and [7].
The approach of [18, 19] is more interested in complexity results and in efficient checking of the satisfiability of equational problems than in computing their solutions. In [18, 19], the set of solutions of an equational problem is expressed by a restricted form of $\exists^*\forall^*$-CNF (called PFE-form), whereas our disjunction of *answers* is a $\exists^*\forall^*$-DNF formula. In order to illustrate that point, we borrow from [18, 19] the following $\exists^*\forall^*$-CNF equational constraint (where $\overline{z}$ and $\overline{y}$ stands for z_1, z_2, z_3, z_4 and y_1, y_2, y_3, y_4):

$$\begin{aligned} \exists \overline{z} \forall \overline{y} [\ (\ & f(z_1, g(z_2)) = f(y_1, z_3) \vee g(y_1) = z_3 \vee f(a, y_2) = f(z_2, g(y_4)) \\ & \vee f(g(y_2), z_1) \neq f(y_3, g(y_1)) \vee g(z_3) \neq z_2\ ) \wedge \\ (\ & f(y_1, a) = f(g(z_2), a) \vee f(g(z_4), y_1) \neq f(z_2, a) \vee g(y_1) \neq g(g(y_3))\ ) \wedge \\ (\ & f(g(z_2), z_1) = f(g(y_1), y_1) \vee g(y_2) = y_3 \vee f(a, y_4) = f(z_2, g(y_2)) \\ & \vee f(f(a, y_3), g(y_2)) \neq f(z_1, y_4) \vee g(z_2) \neq g(f(a, z_3)) \vee z_4 \neq y_1\ )\ ] \end{aligned} \tag{10}$$

In [18, 19], these equational constraints are tranformed (in polinomial time) into the so-called PFE-form. The PFE-form of the (10) is:

$$\begin{aligned} \exists \overline{z} [\ & \exists u_1 \exists u_2 (\ [\ z_1 = g(u_1) \wedge z_2 = g(z_3) \wedge g(u_1) = z_3\ ] \vee \\ & \quad \forall y_1 \forall v [\ z_1 \neq g(y_1) \vee z_2 \neq g(v) \vee z_3 \neq v\ ]\ ) \quad \wedge \\ & (\ [\ z_1 = f(a, z_3) \wedge z_2 = f(a, z_3) \wedge f(g(f(a, z_3)), f(a, u_2)) = f(g(z_4), z_4)\ ] \vee \\ & \quad [\ z_1 = f(a, u_2) \wedge z_2 = f(a, z_3) \wedge f(a, g(a)) = f(z_2, g(a))\ ] \vee \\ & \quad \forall y_3 \forall v (\ z_1 \neq f(a, y_3) \vee z_2 \neq f(a, v) \vee z_3 \neq v)\ )\ ] \end{aligned}$$

The satisfiability of PFE-forms can be checked by a non-deterministic algorithm ([18, 19]) in polinomial-time. Our solver proceeds in a very different way. In particular, we obtain *answers* for constraints with free variables. For the above $\exists^*\forall^*$-CNF constraint (10), our solver first tranforms the inner $\forall^*$-CNF into the disjunction of the following ten *answers* for $\overline{z}$ (for easier reading, each w_i is considered to be existentially quantified):

1. $z_1 = f(a, w_3) \wedge z_2 = w_1 \wedge z_3 = w_2 \wedge \forall v_1 (\ w_3 \neq g(v_1)\ ) \wedge w_1 \neq f(a, w_2)$
2. $z_1 = w_4 \wedge z_2 = w_5 \wedge z_3 = w_6 \wedge \forall v_2 (\ w_4 \neq g(v_2)\ ) \wedge \forall v_3 (\ w_4 \neq f(a, v_3)\ )$
3. $z_1 = f(a, g(w_9)) \wedge z_2 = w_7 \wedge z_3 = w_8 \wedge w_7 \neq f(a, w_8)$

4. $z_1 = w_{10} \wedge z_2 = w_{10} \wedge z_3 = w_{11} \wedge z_4 = w_{10} \wedge \forall v_4(\ w_{10} \neq g(v_4)\ )$
5. $z_1 = g(w_{14}) \wedge z_2 = w_{12} \wedge z_3 = w_{13} \wedge \forall v_5(\ w_{12} \neq g(v_5)\ )$
6. $z_1 = g(w_{15}) \wedge z_2 = g(g(w_{15})) \wedge z_3 = g(w_{15})$
7. $z_1 = g(w_{16}) \wedge z_2 = g(w_{16}) \wedge z_3 = w_{17} \wedge z_4 = g(w_{16}) \wedge \forall v_6(\ w_{16} \neq g(v_6)\ )$
   $\wedge\, w_{17} \neq w_{16}$
8. $z_1 = g(w_{20}) \wedge z_2 = g(w_{18}) \wedge z_3 = w_{19} \wedge \forall v_7(\ w_{18} \neq g(v_7)\ ) \wedge w_{19} \neq w_{18}$
9. $z_1 = g(g(w_{22})) \wedge z_2 = g(g(w_2 2)) \wedge z_3 = w_{21} \wedge z_4 = g(g(w_{22})) \wedge w_{21} \neq g(w_{22})$
10. $z_1 = g(w_{25}) \wedge z_2 = g(g(w_{24})) \wedge z_3 = w_{23} \wedge w_{23} \neq g(w_{24})$

Then, the existencial closure (by $\exists \overline{z} \exists \overline{w}$) of the disjunction of the above ten *answers* is easily reduced to True by checking that each answer is satisfiable. The main goal of [7] is the efficient decidability of equational formulas with a long prefix of quantifiers, focusing on infinite signatures. Because of this focus, they do not deal with the satisfiability test. Besides, the notion of solved-form of [7] allows unrestricted nesting of negation and quantification.

We believe that *answers* could be helpful for development and improvement of resolution- and instance-based methods. On the one hand, for example, the *resolution-based method* presented in [6] can be easily adapted to deal with *answers* instead of $\exists^*\forall^*$-constraints. Moreover, all the $\exists^*\forall^*$-constraints used in the several examples of [6] are really *answers*. With such adjustment, the interesting method of [6] could benefit from compactness (in particular, avoiding explosion rule) and superior performance of basic operations. On the other hand, it seems worth investigating the usefulness of *answers* for the area of growing interest of *instance-based methods*. See [2] for a recent and good summary and for references. In conclusion, we offer some pointers to future developments and applications of the notion of *answer* to the latter area. First, in [12](Sec. 4), it is pointed out that, for redundancy elimination, "*it might be useful to employ elaborate constraint notations such as the ones proposed in [6]*" and *answers* seem to be even better suited for that goal. Second, the notion of *context* used in [3] could be represented by atoms constrained by (disjunction of) *answers*. Then, negation and conjunction of *answers* would become basic for building *context unifiers* and the satisfiability test becomes essential for guiding the procedure.

Acknowledgment: This work has been partially supported by Spanish Projects TIC2001-2476-C03 and TIN2004-079250-C03-03. We also would like to thank the anonymous referees for their valuable comments.

References

1. J. Álvez and P. Lucio. Equational constraint solving using quasi-solved forms. In *Proceedings of the 18th International Workshop on Unification (UNIF'04)*, Cork, Ireland, 5 June 2004.
2. P. Baumgartner and G. Stenz. Instance based methods. *Tutorial T3.- 2nd Int. Joint Conf. on Automated Reasoning, IJCAR 2004, Cork, Ireland*, 2004.
3. P. Baumgartner and C. Tinelli. The Model Evolution Calculus. In Franz Baader, editor, *CADE-19 – The 19th Int. Conf. on Automated Deduction*, volume 2741 of *Lecture Notes in Artificial Intelligence*, pages 350–364. Springer, 2003.

4. W. L. Buntine and H.-J. Bürckert. On solving equations and disequations. *Journal of the ACM*, 41(4):591–629, 1994.
5. Ricardo Caferra and Nicolas Zabel. Extending resolution for model construction. In *JELIA '90: Proceedings of the European workshop on Logics in AI*, pages 153–169, New York, NY, USA, 1991. Springer-Verlag New York, Inc.
6. Ricardo Caferra and Nicolas Zabel. A method for simultaneous search for refutations and models by equational constraint solving. *J. Symb. Comput.*, 13(6):613–641, 1992.
7. 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.
8. H. Comon. Disunification: A survey. In J.L. Lassez and G. Plotkin, editors, *Essays in Honour of Alan Robinson*, 1991.
9. H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425, 1989.
10. Kevin J. Compton and C. Ward Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Ann. Pure Appl. Logic*, 48(1):1–79, 1990.
11. Christian G. Fermüller and Alexander Leitsch. Hyperresolution and automated model building. *J. Log. Comput.*, 6(2):173–203, 1996.
12. H. Ganzinger and K. Korovin. New directions in instantiation-based theorem proving. In *Proc. 18th IEEE Symposium on Logic in Computer Science*, pages 55–64. IEEE Computer Society Press, 2003.
13. Jean-Louis Lassez and Kim Marriott. Explicit representation of terms defined by counter examples. *J. Autom. Reasoning*, 3(3):301–317, 1987.
14. 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. Computer Society Press, 1988.
15. 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.
16. N. Peltier. *Nouvelles Techniques pour la Construction de Modèles finis ou infinis en Déduction Automatique*. PhD thesis, Institut National Polytechnique de Grenoble, 1997.
17. Nicolas Peltier. System description: An equational constraints solver. In *CADE-15: Proceedings of the 15th International Conference on Automated Deduction*, pages 119–123, London, UK, 1998. Springer-Verlag.
18. R. Pichler. Solving equational problems efficiently. In H. Ganzinger, editor, *Proc. 16th Int. Conf. on Automated Deduction (CADE-16)*, number 1632 in Lecture Notes in Artificial Intelligence, pages 97–111. Springer Verlag, 1999.
19. R. Pichler. On the complexity of equational problems in CNF. *Journal of Symbolic Computation*, 36:235–269, 2003.
20. T. Rybina and E. Voronkov. A decision procedure for term algebras with queues. *ACM Trans. Comput. Logic*, 2(2):155–181, 2001.
21. S. Vorobyov. An improved lower bound for the elementary theories of trees. In *Automated Deduction CADE-13 LNAI 110*, pages 275–287. Springer, 1996.

Appendix

In this section, we give the proofs of Theorem 1 and Proposition 1. In the former proof, we make use of some results of [13]. In order to make this paper self-contained, let us recall them using our terminology.

> We consider a fixed term t such that $Var(t) \equiv \overline{x}$ and a fixed collection of substitutions $\theta_1, \ldots, \theta_n$ such that $domain(\theta_j) \equiv \overline{x}$ for every $1 \leq j \leq n$.
> PROPOSITION 4.6. ([13]) *If each $\underline{\mathsf{c}}(\overline{x})\theta_i$ is not a linear term, then there does not exists a finite set of terms equivalent to $t/t\theta_1 \vee \ldots \vee t\theta_n$.* $\square$
> PROPOSITION 4.8. ([13]) *If each $\underline{\mathsf{c}}(\overline{x})\theta_i$ is a linear term, then there exists a finite set of terms equivalent to $t/t\theta_1 \vee \ldots \vee t\theta_n$.* $\square$
> THEOREM 4.1. ([13]) *The algorithm uncover can be used to find an equivalent finite set of terms for $t/t\theta_1 \vee \ldots \vee t\theta_n$ if one exists. Otherwise, uncover will terminate with an implicit representation.* $\square$

Our proof of Theorem 1 is based on the following two lemmas:

Lemma 1. *Let t be a term such that $Var(t) \equiv \overline{x}$ and $\theta_1, \ldots, \theta_n$ be a collection of substitutions such that $domain(\theta_j) \equiv \overline{x}$ and $x_i\theta_j$ is linear for every $1 \leq j \leq n$. $uncover(t, t\theta_1, \ldots, t\theta_n)$ always terminates and yields a (possibly empty) finite set of linear terms that is equivalent to $t/t\theta_1 \vee \ldots \vee t\theta_n$.*

Proof. It is a direct consequence of Proposition 4.8 and Theorem 4.1 in [13]. $\square$

Lemma 2. *Let $\overline{w}^0 \cdot \overline{w}^1$ a disjoint partition of a given tuple $\overline{w}$, σ an assignment with domain $\overline{w}^0$ and $Ext(\overline{w}^1)$ an extension of $\overline{w}^1$ that is infinite for every w_j^1. Let S_0 be a set of UCDs on $\overline{w}$ of the form $\forall \overline{v}(\ w_i \neq s_{ij}(\overline{w}, \overline{v})\ )$ that satisfies the following two properties:*

(P1) *$Var(s_{ij}) \cap \overline{w} \not\equiv \emptyset$ or some v_k appears more than once in s_{ij}*
(P2) *$(\{w_i\} \cup Var(s_{ij})) \cap \overline{w}^1 \not\equiv \emptyset$.*

If every term in $Ext(\overline{w}^1)$ is linear, then σ can be extended to an assignment σ' with domain $\overline{w}$ that satisfies:

$$\sigma \quad \wedge \quad \mathsf{S}_0 \quad \wedge \bigwedge_{w_j^1 \in \overline{w}^1} \ \bigvee_{t \in Ext(w_j^1)} (w_j^1 = t).$$

Proof. Since $\sigma(w_i^0)$ is a ground term $t_i \in \mathcal{T}(\Sigma)$ for each $w_i \in \overline{w}^0$, we can substitute t_i for w_i^0 in S_0. Then, we obtain a finite set S of universal disequations on $\overline{w}^1$ that is trivially equivalent to $\sigma \wedge \mathsf{S}_0$. Besides, by construction, S consists universal disequations of the following three types:

(T1) $\forall \overline{v}(\ w_k^1 \neq s(\overline{v})\ )$ where at least one v_k occurs repeatedly in s
(T2) $\forall \overline{v}(\ w_k^1 \neq s(\overline{w}^1, \overline{v})\ )$
(T3) $\forall \overline{v}(\ t \neq s(\overline{w}^1, \overline{v})\ )$ where t is a ground term and $Var(s) \cap \overline{w}^1 \not\equiv \emptyset$

and σ' should extend σ to $\overline{w}$ (hence, to $\overline{w}^1$) while satisfying:

$$\mathsf{S} \quad \wedge \bigwedge_{w_j^1 \in \overline{w}^1} \bigvee_{t \in Ext(w_j^1)} (w_j^1 = t). \tag{11}$$

Since each $Ext(w_i^1)$ is described by a collection of linear terms, a finite conjunction of universal disequations of type (T1) is not able to turn $Ext(w_i^1)$ up into a finite extension (this follows from Proposition 4.6 in [13]). Besides, each disequation of type (T2) and (T3) involves at least one existential variable with infinite extension. Hence, the finite number of universal disequations of these two types cannot disallow the infinite possible assignments. Therefore, there are infinitely many σ' that satisfies (11). □

Now, we can prove our result:

Theorem 1. *The satisfiability test of Figure 3 terminates for any input* (S, Σ)*. Besides, it returns "satisfiable" if there exists a* Σ*-assignment that satisfies the set* S*. Otherwise, it returns "unsatisfiable".*

Proof. By Lemma 1, the satisfiability test terminates and, moreover, every term in each $Ext(w_i)$ is linear along the whole process. There are four possible exit points:

1. If some $Ext(w_i)$ becomes empty, by Lemma 1, there is not possible assignment with domain w_i that satisfies the subset of S considered at the first step. Hence, there is not assignment for $\overline{w}$ that satisfies S.
2. If $\overline{w}^{fin}$ becomes empty at the end of the first step, then, by Lemma 2, the empty assignment (with empty domain) can be extended to an assignment σ' with domain $\overline{w}$ that satisfies S. Notice that the set of UCDs that satisfies (P1) and (P2) of Lemma 2 are exactly S minus the UCDs considered at the first step.
3. If the second step returns "satisfiable", on the basis of Lemma 1, C contains the finite collection of all possible assignments to $\overline{w}^{fin}$ that satisfies the subset of UCDs involved by the two steps. Therefore, by Lemma 2, each of these possible assignments σ with domain $\overline{w}^{fin}$ can be extended to $\overline{w}$ for satisfying the whole set S.
4. If the second step returns "unsatisfiable" after checking a subset S_0 of UCDs, by Lemma 1, S_0 is unsatisfiable and, hence, S is unsatisfiable too. □

Now, we prove the following result:

Proposition 1. The transformation rule (UD) of Figure 4 is correct.

Proof. (UD) can be obtained by successive applications of the rule (ud_0):

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

where $\overline{v}^1 \equiv Var(t) \cap \overline{v}$ and $\overline{v}^2 \equiv \overline{v} \backslash \overline{v}^1$. It suffices to note that the constraint:

$$\begin{aligned} &\neg\exists\overline{v}^{11}[\ x_1 = t_1(\overline{w}, \overline{v}^{11})\ ] \vee\ \exists\overline{v}^{11}[\ x_1 = t_1(\overline{w}, \overline{v}^{11}) \wedge \neg\exists\overline{v}^{12}(\ x_2 = t_2(\overline{w}, \overline{v}^{12})\ )\ ] \\ &\vee \ldots \vee\ \exists\overline{v}^{11} \ldots \exists\overline{v}^{1(n-1)}[\ x_1 = t_1(\overline{w}, \overline{v}^{11}) \wedge \ldots \wedge x_{n-1} = t_{n-1}(\overline{w}, \overline{v}^{1(n-1)}) \\ &\qquad\qquad\qquad \wedge \neg\exists\overline{v}^{1n}(\ x_n = t_n(\overline{w}, \overline{v}^{1n})\ )\ ] \end{aligned} \tag{12}$$

is equivalent to $\neg\exists\overline{v}^1[\ \overline{x} = \overline{t}(\overline{w}, \overline{v}^1)\ ]$ where $\overline{v}^1 \equiv \overline{v}^{11} \cdot \ldots \cdot \overline{v}^{1n}$. Hence, it only remains to show that the rule (ud_0) is correct. By conjunction (and distribution) of $\neg\exists\overline{v}[\ x = t(\overline{w}, \overline{v}) \wedge \varphi\ ]$ with the tautology $\neg\exists\overline{u}(\ x = t(\overline{w}, \overline{u})\ ) \vee \exists\overline{u}(\ x = t(\overline{w}, \overline{u})\ )$, we obtain two disjuncts. The first disjunct is trivially equivalent to the formula $\neg\exists\overline{v}^1(\ x = t(\overline{w}, \overline{v}^1)\ )$. The second one $\exists\overline{u}(\ x = t(\overline{w}, \overline{u})\ ) \wedge \neg\exists\overline{v}[\ x = t(\overline{w}, \overline{v}) \wedge \varphi\ ]$ is equivalent to:

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

where the quantifier $\forall\overline{v}^1$ has been lifted to the prefix. By distribution and simplification, it yields $\exists\overline{u}\forall\overline{v}^1[\ \overline{u} \neq \overline{v}^1 \vee (x = t(\overline{w}, \overline{u}) \wedge \neg\exists(\ \overline{v}^2\varphi\ )\ )\ ]$. Then, by the well known rule (U_2):

$$(U_2) \qquad \forall\overline{y}(\ P \wedge (y_i \neq t \vee R)\ ) \mapsto\ \forall\overline{y}(\ P \wedge R\{y_i \leftarrow t\}\ )$$

(see, for example, [9]) the constraint is equivalent to:

$$\exists\overline{u}[\ x = t(\overline{w}, \overline{u}) \wedge \neg\exists\overline{v}^2(\ \varphi\{\overline{v}^1 \leftarrow \overline{u}\}\ )\ ].$$

This constraint coincides with the second disjunct in the right-hand side of (ud_0) except for $\overline{v}^1$, which has been renamed $\overline{u}$. □