Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 206 (2008) 23–40
www.elsevier.com/locate/entcs

A New Proposal Of Quasi-Solved Form For Equality Constraint Solving[1]

Javier Álvez[2] Paqui Lucio[3]

Lenguages y Sistemas Informáticos
Basque Country University
San Sebastian, Spain

Abstract

Most well-known algorithms for equational solving are based on quantifier elimination. This technique iteratively eliminates the innermost block of existential/universal quantifiers from prenex formulas whose matrices are in some normal form (mostly DNF). Traditionally used notions of normal form satisfy that every constraint (in normal form) different from *false* is trivially satisfiable. Hence, they are called *solved forms*. However, the manipulation of such constraints require hard transformations, especially due to the use of the distributive and the explosion rules, which increase the number of constraints at intermediate stages of the solving process. On the contrary, *quasi-solved forms* allow for simpler transformations by means of a more compact representation of solutions, but their satisfiability test is not so trivial. Nevertheless, the total cost of checking satisfiability and manipulating constrains using quasi-solved forms is cheaper than using simpler solved forms. Therefore, they are suitable for improving the efficiency of constraint solving procedures. In this paper, we present a notion of quasi-solved form that provides a good trade-off between the cost of checking satisfiability and the effort required to manipulate constraints. In particular, our new quasi-solved form has been carefully designed for efficiently handling conjunction and negation, which are the main Boolean operations necessary to keep matrices of formulas in normal form.

Keywords: free equality theory, term algebra, constraint solving, satisfiability, solved form

1 Introduction

The theory of first order language $\mathcal{L}$ where equality (denoted by $\approx$) is the unique predicate symbol was introduced by Malcev in [12] and it is known as the *Free Equality Theory* (abbreviated by $\mathtt{FET}_{\mathcal{L}}$). Here, we focus on the case for the algebra of finite trees (or Herbrand domain) and finite number of function symbols. The *Free Equality Theory* is axiomatized by the usual equality axioms, the set of axioms E^* (see [3]) and the *Domain Closure Axiom* (DCA), which was first defined in [14] for finite Herbrand domains and extended in [9] for infinite ones. $\mathtt{FET}_{\mathcal{L}}$ was shown

[1] This work has been partially supported by Spanish Project TIN2004-079250-C03-03.
[2] Email: javier.alvez@ehu.es
[3] Email: paqui.lucio@ehu.es

doi:10.1016/j.entcs.2008.03.073

to be decidable in [10] (see also [6]) and, besides, it is also a well-known result that $\mathtt{FET}_{\mathcal{L}}$ is non-elementary (see [7,16]). The inherent complexity of the satisfiability problem of equality constraints (i.e. where the quantifier prefix is of the form $\forall^* \exists^*$) for finite signatures is studied in [13].

The problem of solving equality constraints without negation is known as *unification* and has been widely studied. However, unification problems lack of expressiveness since the equality constraint $\forall v\ (x \not\approx f(v, v))$ cannot be transformed into a finite negation-free first order formula ([8]). Otherwise, if negation is allowed, then the problem is known as *disunification* or (general) equality constraint solving.

Most well-known algorithms for equality constraint solving (see [6,11,12]) and later extensions to richer theories (see [15]) are based on quantifier elimination. This method keeps formulas in some prenex normal form and iterates a procedure that, at each step, eliminates the innermost block of existential/universal quantifiers of the prefix. Most of the decision methods for $\mathtt{FET}_{\mathcal{L}}$ represents formulas in disjunctive or conjunctive normal form, eliminating the innermost block of existential and universal quantifiers respectively. Regarding the first ones, the innermost block of existential quantifiers can be eliminated by simply removing all the equations/disequations involving their variables. However, when the innermost block is universal, double negation is applied in order to turn the block into existential. Hence, these decision methods require the transformation of formulas into disjunctive normal form before and after the elimination of universal quantifiers.

The above easy elimination of the innermost block of existential quantifiers is only correct if each conjunction of equations and disequations is individually satisfiable. Traditional normal forms in equality constraint solving are trivially satisfiable (if different from the constant *false*) and are thus called *solved forms*, but require much effort on transformations. In particular, they require many applications of distribution, which drastically increases the size of formulas. On the contrary, other normal forms facilitate transformations but are not trivially satisfiable, requiring the resolution of hard satisfiability problems. Therefore, a good balance between easy transformations and easy satisfiability checks is highly desirable.

For unification problems, the most commonly used solved form is a finite conjunction of equations defining an idempotent substitution: *simple formulas* (see [5,10]). In order to deal with negation, disunification requires more sophisticated (quasi-)solved forms. We classify them into *existential* and *universal* forms, depending on whether they allow universal quantification or not. Among the existential forms, we find two similar notions: *basic formulas* (see [4,5]) and *definitions with constraints* (see [6]). The satisfiability test on these existential forms is trivial, since any constraint syntactically different from *false* is always satisfiable. However, satisfiability checking on universal forms usually requires the resolution of hard satisfiability problems. The universal form notions of *substitutions with exceptions* and *constrained substitutions* are defined in [2], where the authors introduce a method for solving a *system of equations and disequations* with the proviso that a satisfiability test on *substitutions with exceptions* is given. In [2], they also show 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. However, any *substitution with exceptions* can be easily transformed into a disjunction of quasi-solved forms, as the proposed in this paper, for checking its satisfiability. Other universal forms were already used in [8,10]: *implicit representations* and Boolean combinations of *simple formulas*, respectively.

In this paper, we present a notion of normal form that combines two features: it enables very easy transformations, while testing satisfiability is not hard. Hence, we say that our normal form is a *quasi*-solved form. This notion of quasi-solved form is an improvement of the one introduced in [1]. In particular, it minimizes the application of distribution by means of a more compact representation of solutions. The basic transformations for this new quasi-solved form —conjunction and negation— are very similar to the transformations in [1]. Hence, in this work, we pay special attention to the satisfiability test, which is strongly based on the procedure presented in [8] for transforming implicit representations of sets of terms into explicit.

Outline of the paper. In the next section, we give some notations and preliminary results. Section 3 introduces the notion of quasi-solved form, called `QSNF`, and its basic operations. In Section 4, we describe an efficient satisfiability test on `QSNF`s. Finally, in Section 5, we give some conclusions and discuss future work.

2 Preliminaries

In this section, we recall some basic notions and introduce some notations. For concepts that we do not state here, the reader is referred to [6].

Along the paper, the equality predicate is written using the symbol $\approx$ in order to avoid confusion with the meta-language equality, which is denoted by $=$.

Tuples of objects are denoted using a *bar*. For example, $\overline{x}$ denotes a tuple of variables. Concatenation of tuples is denoted by $\cdot$, that is the expression $\overline{x}\cdot\overline{y}$ denotes the tuple that is obtained from concatenating the elements of $\overline{x}$ and $\overline{y}$. Besides, some classical operations of sets are used for tuples with the obvious meaning ($\cap$, $\cup$, ...). In order to treat tuples of terms and single terms in a uniform way, we use a fresh function symbol $\mathtt{c}$ as tuple constructor. By abuse of notation, $\mathtt{c}$ can construct tuples of any arity $m > 0$. By convenience, we will treat the tuple constructor $\mathtt{c}$ as just another function symbol, but considering that $\mathtt{c} \notin \mathcal{F}_{\mathcal{L}}$.

An *equality constraint* (or, simply, a constraint) is an arbitrary first-order formula consisting of function symbols from a first order language $\mathcal{L}$, equality as the unique predicate symbols and variables from a denumerable set $\mathcal{X}$. As usual, first order formulas are built using the constants *true* and *false*, the connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ and the quantifiers $\exists$, $\forall$. The expressions $\mathtt{Var}(\mathcal{O})$ and $\mathtt{Free}(\mathcal{O})$ respectively denote the variables and the free variables (excluding the quantified ones) occurring in the syntactic object $\mathcal{O}$. Besides, $c^{\exists\setminus\overline{w}}$ and $c^{\forall\setminus\overline{w}}$ are abbreviations for $\exists\overline{v}\ (c)$ and $\forall\overline{v}\ (c)$ respectively, where $\overline{v} = \mathtt{Var}(c) \setminus \overline{w}$.

A term t is said to be *ground* if no variable occurs in t. Besides, t is said to be

linear if there is no variable repetition. The expression $\mathcal{H}_{\mathcal{L}}$ stands for the algebra of all ground terms or Herbrand universe that can be construct using the language $\mathcal{L}$, whereas $\mathcal{T}_{\mathcal{L}}(\mathcal{X})$ denotes the set of all possible terms.

If s and r are terms, then $s \approx r$ is an equation and $s \not\approx r$ is a disequation. Equations and disequations are said to be *collapsing* if at least one of the its terms is a variable. The expressions `EQCE` and `UQCD` are abbreviations for existentially quantified collapsing equation and universally quantified collapsing disequation respectively.

A *substitution* is a mapping from a finite set of variables $\overline{x} \subset \mathcal{X}$, called *domain*, into $\mathcal{T}_{\mathcal{L}}(\mathcal{X})$, called *range*. It is assumed that any substitution behaves as the identity for the variables outside its domain. Given any substitution σ, $\texttt{domain}(\sigma)$ and $\texttt{range}(\sigma)$ respectively denote the domain and range of σ. The composition of substitutions is denoted by juxtaposition, i.e. $\alpha\beta$ denotes the composition of α and β. The *restriction* of a substitution σ to a set of variables $\overline{x}$, denoted by $\sigma_{|\overline{w}}$, is defined as $\{\ (x \leftarrow t) \mid (x \leftarrow t) \text{ and } x \in \overline{x}\ \}$. Besides, a substitution σ is said to be *linear* if $\texttt{range}(\sigma)$ has no repeated variables and σ is said to be an assignment if $\texttt{Var}(\texttt{range}(\sigma)) = \emptyset$.

The most general unifier of a set $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$, denoted by $\texttt{mgu}(\mathcal{O}_1, \ldots, \mathcal{O}_n)$ is an idempotent substitution σ such that $\mathcal{O}_i\sigma = \mathcal{O}_j\sigma$ for every $1 \leq i, j \leq n$ and, for any other substitution θ with the same property, $\theta = \sigma\alpha$ for some substitution α. If $\texttt{mgu}(\mathcal{O}_1, \ldots, \mathcal{O}_n)$ does (not) exist, then the objects $\mathcal{O}_1, \ldots, \mathcal{O}_n$ are said to be *(non-)unifiable*. Besides, the most general common instance of a unifiable set of objects $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$, denoted by $\texttt{mgi}(\mathcal{O}_1, \ldots, \mathcal{O}_n)$, is $\mathcal{O}_i\sigma$ for any $1 \leq i \leq n$, where $\sigma = \texttt{mgu}(\mathcal{O}_1, \ldots, \mathcal{O}_n)$.

An assignment σ is said to be a *solution* of the constraint c if $\texttt{domain}(\sigma) \subseteq \texttt{Free}(c)$ and $\texttt{FET}_{\mathcal{L}} \models c\sigma$. A constraint c is said to be *satisfiable* if it has at least one solution, and the constraint c is said to be *non-satisfiable* otherwise. Besides, two constraints c_1 and c_2 are said to be *equivalent* if, for every assignment σ, σ is a solution of c_1 iff σ is a solution of c_2.

The *characteristic term* of an equation $w \approx r$ or a disequation $w \not\approx r$ w.r.t. a set of variables $\overline{w}$ such that $w \in \overline{w}$ is the term $\texttt{c}(\overline{w})\sigma$ where

$$\sigma = (\theta \cup \{w \leftarrow r\theta\})$$

$\overline{w}' = \overline{w} \setminus \{w\}$ and $\theta = \{\overline{w}' \leftarrow \overline{z}\}$ for some fresh $\overline{z}$. For example, the characteristic term of $w_1 \approx f(w_2, f(v, v))$ is

$$\texttt{c}(\overline{w})\{w_1 \leftarrow f(z, f(v, v)), w_2 \leftarrow z\}$$

where $\overline{w} = w_1 \cdot w_2$. Note that the equations $(\ w_1 \approx f(w_2, f(v, v))\ )^{\exists\setminus\overline{w}}$ and $(\ \texttt{c}(\overline{w}) \approx \texttt{c}(\overline{w})\{w_1 \leftarrow f(z, f(v, v)), w_2 \leftarrow z\}\ )^{\exists\setminus\overline{w}}$ (resp. the disequations $(\ w_1 \not\approx f(w_2, f(v, v))\ )^{\forall\setminus\overline{w}}$ and $(\ \texttt{c}(\overline{w}) \not\approx \texttt{c}(\overline{w})\{w_1 \leftarrow f(z, f(v, v)), w_2 \leftarrow z\}\ )^{\forall\setminus\overline{w}}$) are equivalent. Besides, $\texttt{ChT}(\varphi, \overline{w})$ denotes the set of characteristic terms $\{\texttt{c}(\overline{w})\sigma_1, \ \ldots, \ \texttt{c}(\overline{w})\sigma_n\}$ w.r.t. $\overline{w}$ that is obtained from the disjunction (resp. conjunction) of n `EQCE`s (resp. `UQCD`s) φ.

3 A New Notion of Quasi-Solved Form

In this section, we introduce a new notion of quasi-solved form, abbreviated by QSNF, and its basic operations: that is, the transformation from a conjunction of QSNF*s* or a negated QSNF into an equivalent disjunction of QSNF*s*. Besides, we compare the above transformations to the ones for basic formulas, which is the simplest notion of solved form for unification problems.

Whereas basic formulas simply consist in a conjunction of equations and disequations, QSNFs are formally defined as follows.

Definition 3.1 Let $\overline{x} \subseteq \mathcal{X}$ be a tuple of pairwise distinct variables. A QSNF *for the variables* $\overline{x}$ is either an atom *true*/*false* or a formula $\exists \overline{w}\ (\ a(\overline{x}, \overline{w})\ )$, where $a(\overline{x}, \overline{w})$ is a conjunction of the form

$$\overline{x} \approx \overline{t}\ \wedge\ \bigwedge_{i=1}^{n} [\ \bigwedge_{j=1}^{l_i} (\ w_i \not\approx r_{ij}\ )^{\forall \setminus \overline{w}}\ \wedge\ \bigvee_{k=1}^{o_i} (\ w_i \approx s_{ik}\ )^{\exists \setminus \overline{w}}\ ]$$

such that

- $n,\ l_1,\ \ldots,\ l_n \geq 0$,
- $o_1,\ \ldots,\ o_n \geq 1$,
- the set of variables $\overline{w} = \mathtt{Var}(\overline{t})$ is disjoint from $\overline{x}$,
- $r_{ij} \in \overline{w}$ if r_{ij} is a variable,
- each s_{ik} is a linear term such that $(\overline{w} \cap \mathtt{Var}(s_{ik})) = \emptyset$.

Besides, the variables $\overline{w}$ are called *auxiliary variables*. □

Note that every disjunction of equations on existential variables is non-empty. Otherwise, such disjunction (and, hence, the QSNF) would be equivalent to *false*. Besides, each variable in a term s_{ij} is existential and only occurs in s_{ij}. Hence, a disjunction of equations on an auxiliary variable w may consist of a single equation $\exists z\ (w \approx z)$, which is equivalent to *true*. For technical convenience, we assume that there are always m equations in every QSNF for any m-tuple of variables $\overline{x}$ and, hence, we add a UQCD of the form $x \approx w$ if x does not occur in the QSNF , where w is a fresh variable from $\mathcal{X}$.

There are two main differences between QSNFs and basic formulas: on one hand, QSNFs allow for a restricted kind of universal quantification. On the other hand, QSNFs are not pure DNF formulas because of the disjunctions of equations on auxiliary variables. By means of these two features, QSNFs obtain a more compact representation of solutions than basic formulas, which (as we will show) increases the performance of constraint solving. The next two examples show the compactness of QSNFs for representing solutions.

Example 3.2 Let $\mathcal{F}_{\mathcal{L}} = \{a_{/0},\ g_{/1},\ f_{/2}\}$. The QSNF

$$\exists w\ (\ x \approx f(w, a) \wedge \forall v\ (w \not\approx g(g(v)))\ \wedge\ \forall \overline{v}\ (w \not\approx f(f(v_1, v_2), g(v_3)))\ )$$

Rule (UD)

$$\neg\exists\overline{v}\ (\ \overline{x} \approx \overline{t} \wedge \varphi\ )\ \mapsto\ \neg\exists\overline{v}^1\ (\ \overline{x} \approx \overline{t}\ )\ \vee\ \exists\overline{v}^1\ (\ \overline{x} \approx \overline{t} \wedge \neg\exists\overline{v}^2\ \varphi\ )$$

where $\overline{v}^1 = (\mathtt{Var}(\overline{t}) \cap \overline{v})$ and $\overline{v}^2 = (\overline{v} \setminus \overline{v}^1)$

Rule Explosion

$$\forall\overline{y}\ (\ P\ )\ \mapsto\ \bigvee_{f\in\mathcal{F}_{\mathcal{L}}} \exists\overline{z}\forall\overline{y}\ (\ P \wedge x \approx f(\overline{z})\ )$$

where $x \in \mathtt{Var}(P)$

Fig. 1. Transformation Rules Involving Universal Quantification

is equivalent to the following disjunct of 7 basic formulas.

$$(x \approx f(a,a))\ \vee\ (x \approx f(g(a),a))\ \vee\ \exists\overline{y}\ (x \approx f(g(f(y_1,y_2)),a)) \vee$$

$$(x \approx f(f(a,a),a))\ \vee\ \exists\overline{y}\ (x \approx f(f(a,f(y_1,y_2)),a)) \vee$$

$$\exists y\ (x \approx f(f(g(y),a),a))\ \vee\ \exists\overline{y}\ (x \approx f(f(g(y_1),f(y_2,y_3)),a)).$$ □

Example 3.3 Let $\mathcal{F}_{\mathcal{L}} = \{a_{/0},\ g_{/1},\ f_{/2}\}$. The QSNF

$$\exists w\ (\ x \approx f(w_1,w_2) \wedge w_1 \not\approx w_2 \wedge$$

$$[\ \exists\overline{z}\ (w_1 \approx g(f(z_1,z_2)))\ \vee\ \exists\overline{z}\ (w_1 \approx f(g(z_1),z_2))\ ] \wedge$$

$$[\ \exists\overline{z}\ (w_2 \approx g(f(z_1,z_2)))\ \vee\ \exists\overline{z}\ (w_2 \approx f(z_1,g(z_2)))\ ]\ )$$

is equivalent to the following disjunct of 6 basic formulas

$$\exists\overline{y}\ (x \approx f(g(f(y_1,y_2)),g(f(y_3,y_4)))) \wedge y_1 \not\approx y_3) \vee$$

$$\exists\overline{y}\ (x \approx f(g(f(y_1,y_2)),g(f(y_3,y_4)))) \wedge y_2 \not\approx y_4) \vee$$

$$\exists\overline{y}(x \approx f(g(f(y_1,y_2)),f(y_3,g(y_4)))) \vee$$

$$\exists\overline{y}(x \approx f(f(g(y_1),y_2)),g(f(y_3,y_4))) \vee$$

$$\exists\overline{y}(x \approx f(f(g(y_1),y_2),f(y_3,g(y_4))) \wedge y_2 \not\approx g(y_4)) \vee$$

$$\exists\overline{y}(x \approx f(f(g(y_1),y_2),f(y_3,g(y_4))) \wedge y_3 \not\approx g(y_1)).$$ □

QSNFs have been proposed for improving the efficiency of general equality constraint solving methods based on the quantifier elimination technique. For solving any formula, these methods keep, at each step, the matrix of the formula as a disjunction of the form $\bigvee_{i=1} a_i$, where each a_i is a satisfiable constraint in some quasi-solved form. If the innermost block of quantifiers is existential, then it can be easily eliminated. However, when the innermost block is universal, double negation

is applied as follows

$$\neg \bigvee_{i=1}^{n} a_i \longmapsto \bigwedge_{i=1}^{n} \neg a_i \overset{(i)}{\longmapsto} \bigwedge_{i=1}^{n} \bigvee_{j=1}^{l_i} b_{ij} \longmapsto \bigvee_{j=1}^{m} \bigwedge_{k=1}^{n_j} b_{jk} \overset{(ii)}{\longmapsto} \bigvee_{j=1}^{m} \bigvee_{h=1}^{o} c_{jh}$$

involving both (i) negation and (ii) conjunction on the selected quasi-solved form. The notion of QSNFs has been designed to improve the above transformation by means of its two main features. On one hand, disjunctions of equations on auxiliary variables minimize the use of distribution in many cases. On the other hand, universal quantification drastically reduces the use of the explosion rule by means of the rule (UD) (see Figure 1). Roughly speaking, the rule (UD) splits a universally quantified formula into two parts, turning universal quantification into existential in the second subformula. Thus, by successive applications, the scope of each universal quantifier can be restricted to a single disequation. Hence, the rule (UD) is extensively used for both transformations on QSNFs: conjunction and negation. Besides, the two subformulas that are obtained by the rule (UD) are complementary, which is an important feature regarding conjunction: that is, because of the combination of negation and conjunction in the quantifier elimination technique, we often often has to simplify a conjunction of disjunctions of QSNFs that have been obtained by application of the rule (UD). Therefore, the complementary nature of the QSNFs in each disjunction easily reduces to *false* many of the combinations that are obtained by distribution.

In the next two subsections, we claim that negation and conjunction operations on QSNFs can be efficiently performed using some examples. Besides, we compare these operations on QSNFs with the same operations on basic formulas.

3.1 Negation

The syntactic form of QSNFs has been especially designed to deal with negation in a efficient way. The main advantages of using QSNFs for dealing with negation are shown in the next example:

$$\neg\exists\overline{w}\ (\ x \approx f(w_1, w_2) \wedge \forall v\ (w_1 \not\approx g(f(v, v)))\ \wedge\ \forall v\ (w_1 \not\approx f(a, v)) \wedge$$
$$\forall v\ (w_1 \not\approx f(v, a))\ \wedge\ [\ (w_2 \approx a) \vee \exists\overline{z}\ (w_2 \approx f(z_1, z_2))\ ]\ )$$

On one hand, the restricted kind of universal quantification in QSNFs provides us an straightforward method for transforming the negated conjunction of equations into a disjunction of QSNFs, using the rule (UD). That is, the application of (UD) in the above negated QSNF yields

(1) $\neg\exists\overline{w}\ (x \approx f(w_1, w_2))\ \vee$

(2) $\exists\overline{w}\ (\ x \approx f(w_1, w_2)\ \wedge\ \neg(\ \forall v\ (\ w_1 \not\approx f(v, v)\ )\ \wedge\ \forall v\ (w_1 \not\approx f(a, v))\ \wedge$
$\forall v\ (w_1 \not\approx f(v, g(a)))\ \wedge\ [\ (w_2 \approx a) \vee \exists\overline{z}\ (w_2 \approx f(z_1, z_2))\ ]\ )$

where the subformula (1) is already a QSNF for the variable x

(3) $\exists w\ (\ x \approx w \wedge \forall\overline{v}\ (w \not\approx f(v_1, v_2))\ ).$

Therefore, the first part of QSNFs can be easily negated.

Regarding the second subformula, negated disjunctions of equations are trivially transformed into conjunctions of UQCDs and, besides, negated conjunctions of UQCDs are turned into disjunctions of equations. Hence, we only need to split each disjunction of equations according to its characteristic terms. In our example, the only UQCD in subformula (2) with a non-linear characteristic term is $\forall v\ (\ w_1 \not\approx f(v,v)\ )$, thus its negation $\exists v\ (\ w_1 \approx f(v,v)\ )$ is distributed over the subformula. After simplification, we already obtain a QSNF for the variable x

$$\exists w_2 \cdot v\ (\ x \approx f(f(v,v),w_2)\ ). \tag{4}$$

Besides, just by distribution, we also get the following two QSNFs :

$$\exists \overline{w}\ (\ x \approx f(w_1,w_2)\ \wedge\ [\ \exists v\ (w_1 \approx f(a,v))\ \vee\ \exists v\ (w_1 \approx f(v,g(a)))\ ]\ ) \tag{5}$$

$$\exists \overline{w}\ (\ x \approx f(w_1,w_2)\ \wedge\ (w_2 \not\approx a)\ \wedge\ \forall \overline{z}\ (w_2 \approx f(z_1,z_2))\ ) \tag{6}$$

Hence, the result of negating the initial QSNF is the disjunction consisting of (3), (4), (5) and (6). Note that transformation of this disjunction into a disjunction of basic formulas would also require several applications of the explosion rule in QSNFs (3) and (6), and also distributing the disjunction of EQCEs on v of (5).

3.2 Conjunction

Conjunction requires some basic transformation steps that are common for every (quasi-)solved form consisting in a conjunction of equations and disequations. After these common transformations, conjunction require more transformations according to the characteristics of each (quasi-)solved form. In this subsection, we will show these transformation steps by means of the following example. Let $\mathcal{F}_{\mathcal{L}} = \{a_{/0},\ g_{/1},\ f_{/2}\}$, we are going to transform the conjunction of

$$\exists \overline{y}\ (\ x_1 \approx y_1 \wedge x_2 \approx f(y_1,y_2) \wedge \forall v\ (y_1 \not\approx f(a,v)) \wedge \forall \overline{v}\ (y_1 \not\approx g(f(v_1,v_2)))\ \wedge \tag{7}$$

$$[\ \exists z\ (y_1 \approx f(z,g(a))) \vee \exists \overline{z}\ (y_1 \approx f(f(g(z_1),z_2),z_3))\ ]\ \wedge \tag{8}$$

$$\forall v\ (y_2 \not\approx f(v,a)) \wedge \forall v\ (y_2 \not\approx g(v))\ \wedge \tag{9}$$

$$[\ \exists \overline{z}\ (y_2 \approx f(f(z_1,a),z_2)) \vee \exists \overline{z}\ (y_2 \approx f(g(z_1),z_2))\ ]\ ) \wedge \tag{10}$$

with the following QSNF

$$\exists \overline{w}\ (\ x_1 \approx f(w_1,a) \wedge x_2 \approx w_2 \wedge \forall \overline{v}\ (w_1 \not\approx f(g(v_1),v_2)) \wedge w_1 \not\approx g(a)\ \wedge \tag{11}$$

$$[\ \exists \overline{z}\ (w_1 \approx f(z_1,f(z_2,z_3))) \vee \exists \overline{z}\ (\ w_1 \approx g(f(z_1,z_2))\ )\ ]\ \wedge \tag{12}$$

$$\forall v\ (w_2 \not\approx f(a,v)) \wedge \forall \overline{v}\ (w_2 \not\approx f(v_1,g(v_2)))\ \wedge \tag{13}$$

$$[\ \exists z\ (w_2 \approx f(g(z_1),a)) \vee \exists \overline{z}\ (w_2 \approx f(f(f(z_1,z_2),z_3),z_4))\ ]\ ) \tag{14}$$

into an equivalent disjunctions of QSNF*s*. This transformation is easily generalizable conjunctions of n QSNFs for arbitrary $n \geq 2$.

First, we check if the equational parts of the QSNFs unify. In our example, the most general unifier is $\sigma = \{y_1 \leftarrow f(w_1,a),\ w_2 \leftarrow f(f(w_1,a),y_2)\}$. In case the most general unifier does not exist, the whole conjunction is reduced to *false*. Otherwise, the formula is transformed into the most general instance of the equational parts

in conjunction with all the disequational parts, applying the most general unifier of the equational parts.

Then, the next transformation steps depend on the particular features of each (quasi-)solved form. Dealing with basic formulas and since their disequational part consists of existentially quantified disequations, the next transformations combine disunification and distribution. For QSNFs, the treatment of UQCDs is almost the same since universal quantification does not affect to disunification and distribution is possible by means of the rule (UD). Further, the formulas that are obtained from most general unifiers are simpler in the case of UQCDs. That is, every mapping from a universal variable to any term can be removed, because such mappings yield disequations that are trivially equivalent to *false*. Such a transformation is applied to the first UQCD in (7):

$$\begin{aligned} \forall v\ (y_1 \not\approx f(a,v))\sigma \ &= \ \forall v\ (f(w_1,a) \not\approx f(a,v)) \mapsto (\ w_1 \not\approx a \vee \forall v\ (v \not\approx a)\ ) \\ &\mapsto (w_1 \not\approx a) \end{aligned} \tag{15}$$

Besides, when the most general unifier does not exist, the UQCD is replaced by *true*, as it occurs with the second UQCD in (7):

$$\forall \overline{v}\ (y_1 \not\approx g(f(v_1,v_2)))\sigma \ = \ \forall \overline{v}\ (f(w_1,a) \not\approx g(f(v_1,v_2))) \ \mapsto \ \mathit{true}$$

Unlike basic formulas, QSNFs allow disjunctions of equations in the second part of the formula, which have to be transformed combining unification and distribution. In this way, the disjunction of EQCEs in (8) is transformed as follows:

$$\begin{aligned} \exists z\ (y_1 \approx f(z,g(a)))\sigma \ \vee \ \exists \overline{z}\ (\ y_1 \approx f(f(g(z_1),z_2),z_3)\ )\sigma \\ = \ \exists z\ (f(w_1,a) \approx f(z,g(a))) \ \vee \ \exists \overline{z}\ (\ f(w_1,a) \approx f(f(g(z_1),z_2),z_3)\ ) \\ \mapsto \ \mathit{false} \ \vee \ \exists \overline{z}\ (w_1 \approx f(g(z_1),z_2) \wedge z_3 \approx a) \\ \mapsto \ \exists \overline{z}\ (w_1 \approx f(g(z_1),z_2)) \end{aligned} \tag{16}$$

Since $(\mathtt{domain}(\sigma) \cap \{y_2,\ w_1\}) = \emptyset$, the UQCDs in (9, 11) and the disjunctions of EQCEs in (10, 12) do not change. However, σ affects to the UQCDs in (13) and the EQCEs in (14). In the former, the first UQCD in (13) is replaced with *true* since the most general unifier does not exist:

$$\forall v\ (\ w_2 \not\approx f(a,v)\ )\sigma \ = \ \forall v\ (\ f(f(w_1,a),y_2) \not\approx f(a,v)\ ) \ \mapsto \ \mathit{true}$$

Whereas the second UQCD in (13) is transformed in the following way:

$$\begin{aligned} \forall \overline{v}\ (w_2 \not\approx f(v_1,g(v_2)))\sigma \ &= \ \forall \overline{v}\ (f(f(w_1,a),y_2) \not\approx f(v_1,g(v_2))) \\ &\mapsto \forall \overline{v}\ (v_1 \not\approx f(w_1,a) \vee y_2 \not\approx g(v_2)) \\ &\mapsto \forall v_2\ (y_2 \not\approx g(v_2)) \end{aligned} \tag{17}$$

In the latter, the disjunction of EQCEs in (14) is transformed as follows:

$$\begin{aligned} \exists z\ (w_2 \approx f(g(z_1),a))\sigma \ \vee \ \exists \overline{z}\ (w_2 \approx f(f(f(z_1,z_2),z_3),z_4))\sigma \\ \mapsto \ \exists z\ (f(f(w_1,a),y_2) \approx f(g(z_1),a)) \ \vee \\ \exists \overline{z}\ (f(f(w_1,a),y_2) \approx f(f(f(z_1,z_2),z_3),z_4)) \\ \mapsto \ \mathit{false} \ \vee \ \exists \overline{z}\ (w_1 \approx f(z_1,z_2) \wedge z_3 \approx a \wedge y_2 \approx z_4) \end{aligned}$$

(18) $\quad \mapsto \exists\overline{z}\,(w_1 \approx f(z_1, z_2))$

Finally, the conjunction of disjunctions of EQCEs on the same variables have to be transformed into a single disjunction. In our example, there is only one disjunction of EQCEs on y_2 (subformula (10)), but we have a conjunction of three disjunctions of EQCEs on w_1 (subformulas (16, 12, 18)), which is equivalent to

(19) $\quad \exists\overline{z}\,(\, w_1 \approx f(g(z_1), f(z_2, z_3))\,).$

Therefore, the initial conjunction of two QSNFs is equivalent to: [4]

$$\exists\overline{y}\,(\, x_1 \approx f(w_1, a) \wedge x_2 \approx f(f(w_1, a), y_2) \wedge$$

(11) $\quad \forall\overline{v}\,(w_1 \not\approx f(g(v_1), v_2)) \;\wedge\; w_1 \not\approx g(a) \wedge$

(15, 19) $\quad w_1 \not\approx a \;\wedge\; [\,\exists\overline{z}\,(w_1 \approx f(g(z_1), f(z_2, z_3)))\,] \wedge$

(9) $\quad \forall v\,(y_2 \not\approx f(v, a)) \;\wedge\; \forall v\,(y_2 \not\approx g(v)) \wedge$

(10) $\quad [\,\exists\overline{z}\,(y_2 \approx f(f(z_1, a), z_2)) \;\vee\; \exists\overline{z}\,(y_2 \approx f(g(z_1), z_2))\,]\,)$

Note that, since QSNFs includes disjunctions of EQCEs on auxiliary variables, the use of distribution can be delayed during the solving process, only performing the transformations that are needed to keep the constraint in normal form.

4 Satisfiability Test on QSNFs

In exchange for compact representation, a QSNF syntactically different from *false* may be non-satisfiable. That is, given a QSNF with auxiliary variables $\overline{w}$, it may occurs that no assignment of domain $\overline{w}$ satisfies the conjunction of UQCDs and disjunctions of EQCEs. Hence, checking the satisfiability on QSNFs (or checking the existence of such an assignment) has to be easy in order to efficiently solving general equality constraints.

In this section, we give a procedure a procedure for checking the satisfiability of a QSNF. This test is strongly based on the algorithm *uncover* (see Figure 2) that was introduced in [8], where the authors study the notions of *explicit* and *implicit* representations of sets of terms. An *explicit representation* of a set of terms simply consists of a finite disjunction of terms, whereas an *implicit representation* of a set of terms is an expression $t/\{t_1 \vee \ldots \vee t_n\}$ that represents those ground instances of t that are not instances of t_i for any $1 \leq i \leq n$. Obviously, an explicit representation is also an implicit representation. However, implicit representations cannot be always transformed into an explicit representation. For example, being $\mathcal{F}_{\mathcal{L}} = \{a_{/0}, b_{/0}, f_{/1}\}$, the implicit representation

$$\mathsf{c}(w_1, w_2)/\{\mathsf{c}(v_1, a) \vee \mathsf{c}(b, v_2) \vee \mathsf{c}(v_3, g(v_4))\}$$

is refined to the explicit one $\{\mathsf{c}(a, b), \mathsf{c}(g(z), b)\}$, whereas $\mathsf{c}(w_1, w_2)/\{\mathsf{c}(v, v)\}$ cannot be explicitly represented.

[4] Note that the second UQCD in (9) and the one in (17) are equal modulo renaming.

uncover$(t, t\theta_1, \ldots, t\theta_n)$ `is`

`if` $\exists\, t\theta_i$ such that θ_i is linear `then`
$P := partition(t, \theta_i)$
`return` $\bigcup_{r \in P} uncover(r, t'_1, \ldots, t'_{i-1}, t'_{i+1}, \ldots, t'_n)$
`where` $t'_j = \mathtt{mgi}(r, t\theta_j)$ for each $1 \leq j \leq i-1$ and $i+1 \leq j \leq n$
`else return` $t/\{t\theta_1 \vee \ldots \vee t\theta_n\}$
`end if`

partition(t, θ) `is`

`if` θ is a renaming `then return` $\emptyset$
`else`
select $\{w \leftarrow f(s_1, \ldots, s_m)\} \in \theta$
$r := t\{w \leftarrow f(v_1, \ldots, v_m)\}$
$\sigma := \theta \setminus \{w \leftarrow f(s_1, \ldots s_m)\} \cup \bigcup_{i=1}^{m} \{v_i \leftarrow s_i\}$
where $v_1, \ldots, v_m$ are fresh variables
`return` $partition(r, \sigma) \cup \bigcup_{g \in \mathcal{F}_{\mathcal{L}}, g \neq f} t\{z \leftarrow g(\overline{z})\}$
where $\overline{z}$ are fresh variables
`end if`

Fig. 2. The algorithm *uncover*

In [8], the authors propose the algorithm *uncover* that decides if an implicit representation can be transformed into an explicit one, returning an explicit representation or an irreducible implicit representation respectively. The algorithm *uncover* decides on the basis of the following two results from [8].

Proposition 4.1 *If the substitution θ_i is not linear for some $1 \leq i \leq n$, then the implicit representation $t/\{t\theta_1 \vee \ldots \vee t\theta_n\}$ has no equivalent explicit representation.*

Proof. See Proposition 4.6 in [8].

Proposition 4.2 *If the substitution θ_i is linear for every $1 \leq i \leq n$, then the implicit representation $t/\{t\theta_1 \vee \ldots \vee t\theta_n\}$ has an equivalent explicit representation.*

Proof. See Proposition 4.8 in [8].

The original formulation in [8] is based on the notion of *restriction of an instance $t\theta$ w.r.t. the term t*. That is, an instance $t\theta$ is said to be *restricted w.r.t. the term t* if any variable appears more than once in the terms $x_1\theta, \ldots, x_l\theta$, where $\mathtt{Var}(t) = \{x_1, \ldots, x_l\}$. For example, $f(x_1, x_2)\{x_1 \leftarrow x_2\}$ is restricted w.r.t. $f(x_1, x_2)$ since the variable x_2 appears twice in the terms $x_1\{x_1 \leftarrow x_2\}$ and $x_2\{x_1 \leftarrow x_2\}$. Under the proviso that $\mathtt{domain}(\theta) = \mathtt{Var}(t)$ (and therefore $(\mathtt{range}(\theta) \cap \mathtt{Var}(t)) = \emptyset$), an instance $t\theta$ is restricted w.r.t. the term t iff θ is non-linear. That proviso is always reachable

since, given any instance $t\theta$, we can find a substitution θ' such that $t\theta'$ is a renaming of $t\theta$ and $\mathtt{domain}(\theta') = \mathtt{Var}(t)$: we just need to transform θ into $\beta \cup \{x \leftarrow z\}$ for each variable $x \in (\mathtt{Var}(t) \cap \mathtt{range}(\theta))$, where $\mathtt{domain}(\beta) = \mathtt{domain}(\theta)$, $\mathtt{range}(\beta) = \mathtt{range}(\theta)\{x \leftarrow z\}$ and z is a fresh variable from $\mathcal{X}$. For example, this transformation yields $f(x_1, x_2)\{x_1 \leftarrow z, x_2 \leftarrow z\}$ for the above instance $f(x_1, x_2)\{x_1 \leftarrow x_2\}$. From now on, we will assume that $\mathtt{domain}(\theta) = \mathtt{Var}(t)$ for any instance $t\theta$ and, therefore, we use the results of [8] but testing linearity instead of checking if instances are restricted.

For our purposes, we next redefine the notion of explicit representation for tuples of variables.

Definition 4.3 An *explicit representation for an m-tuple of variables $\overline{x}$* is a set of linear terms of the form $\mathtt{c}(s_1, \ldots, s_m)$ such that $s_i \in \mathcal{T}_{\mathcal{L}}(\mathcal{X})$ for each $1 \leq i \leq m$. □

Intuitively, explicit representations use variables for finitely representing an infinite universe of ground terms. Hence, the ground/non-ground nature of terms in an explicit representation relates to the finite/infinite nature of the represented universe of ground terms. Explicit representations are denoted by capital Greek letter Δ, possibly with sub-scripts, followed by the tuple of variables between brackets. For example, $\Delta[x_1, x_2, x_3]$, $\Delta_0[x_2, x_3]$ and $\Delta_1[x_1, x_3]$ are three explicit representations, each one for a different tuple of variables.

Besides, we introduce two operations on explicit representations. The first operation is the projection of an explicit representation to a subset of its variables.

Definition 4.4 Let $\Delta[\overline{x}]$ be an explicit representation for the n-tuple of variables $\overline{x}$ and $\overline{z} \subseteq \overline{x}$ a m-tuple of variables. The *projection of the explicit representation $\Delta[\overline{x}]$ to $\overline{z}$*, denoted by $\Delta[\overline{z}]$, consists of a term $\mathtt{c}(s_{i_1}, \ldots, s_{i_m})$ for each $\mathtt{c}(s_1, \ldots, s_n) \in \Delta[\overline{x}]$ such that every $x_{i_j} \in \overline{z}$. □

The second operation is the Cartesian product of two explicit representations for pairwise disjoint tuples of variables. The generalization to the Cartesian product of n explicit representations for pairwise disjoint tuples of variables is trivial.

Definition 4.5 Let $\overline{x}$ and $\overline{z}$ respectively be an n-tuple and an m-tuple of variables such that $\overline{x} \cap \overline{z} = \emptyset$. The *Cartesian product of the explicit representations $\Delta_1[\overline{x}]$ and $\Delta_2[\overline{z}]$*, denoted by $\Delta_1[\overline{x}] \times \Delta_2[\overline{z}]$, is the explicit representation that consists of all the terms of the form $\mathtt{c}(s_1, \ldots, s_n, r_1, \ldots, r_m)$ such that $\mathtt{c}(s_1, \ldots, s_n) \in \Delta_1[\overline{x}]$ and $\mathtt{c}(r_1, \ldots, r_m) \in \Gamma_2[\overline{z}]$. □

As a initial approach to the QSNFs satisfiability test, we first explain a very simple method using the algorithm *uncover*: let us consider a QSNF of the form

$$\exists \overline{w} \, (\, \overline{x} \approx \overline{t} \, \wedge \bigwedge_{i=1}^{n} [\, \varphi_i \wedge \psi_i \,] \,)$$

where $\overline{w}$ is the n-tuple of auxiliary variables, each φ_i is a conjunction of UQCDs and each ψ_i is a disjunction of EQCEs such that its characteristic term is linear. The

set_uncover$(\{t_1, \ldots, t_m\}, \{s_1, \ldots, s_n\})$`is`

`for each` $t_i \in \{t_1, \ldots, t_m\}$ `do`

$U_i := uncover(t_i, r_i^1, \ldots, r_i^n)$ where each $r_i^j = \texttt{mgi}(t_i, s_j)$

`end do`

`return` $\bigcup_{i=1}^{m} U_i$

Fig. 3. A Generalization of *uncover* for Sets

method proceeds in two steps. First, we obtain an explicit representation for the variables $\overline{w}$ from $\bigwedge_{i=1} \psi_i$ in the following way: $\Delta_0[\overline{w} =] = \Delta_0[w_1] \times \ldots \times \Delta_0[w_n]$, where $\Delta_0[w_i] = \texttt{ChT}(w_i, \psi_i)$ for each $1 \leq i \leq n$. Note that the characteristic term of every `EQCE` in ψ_i is linear. Second, $\Delta_0[\overline{w}]$ is refined using the algorithm *uncover* according to the `UQCD`s. For simplicity, we provide the algorithm *set_uncover* (see Figure 3) that conveniently invokes *uncover*. Thus, all the calls to *uncover* can be summed up as $set\_uncover(\Delta_0[\overline{w}], \texttt{ChT}(\overline{w}, \bigwedge_{i=1}^{n} \varphi_i))$. The call to *set_uncover* returns an explicit or an implicit representation, so the given `QSNF` is decided to be satisfiable unless *set_uncover* returns an empty explicit representation. Note that, when *set_uncover* returns an implicit representation, there exist infinitely many assignments that satisfy the given `QSNF` , which cannot be finitely enumerated.

The correctness of the above satisfiability test is directly given by the following result.

Theorem 4.6 *The algorithm uncover finds an equivalent explicit representation for the implicit representation* $t/\{t\theta_1 \vee \ldots \vee t\theta_n\}$ *if one exists. Otherwise, the algorithm uncover terminates with an implicit representation.*

Proof. See Theorem 4.1 in [8].

This initial approach to the satisfiability test uses all the `UQCD`s in the given `QSNF` to refine the explicit representation as much as possible, and refinement is a very time-consuming task. However, many of the `UQCD`s in a `QSNF` may not affect satisfiability. Hence, the satisfiability test on `QSNF`s can be optimized by exclusively performing the refinements that are strictly necessary to decide satisfiability. This is the underlying idea of our `QSNF` satisfiability test, which is described in Figure 4.

Our refined test also works in two steps. At the first step, the conjunction of `UQCD`s on each auxiliary variable φ_i is split into two conjunctions, separating the `UQCD`s with a linear characteristic term from the ones with a non-linear characteristic term into φ_i^1 and φ_i^2 respectively. Then, the initial explicit representation for each auxiliary variable, which is obtained from its disjunction of `EQCE`s as before, is refined according to the conjunction of `UQCD`s with a linear characteristic term. Linearity ensures that *set_uncover* returns an empty or non-empty explicit representation (see Proposition 4.2). If the returned explicit representation for some auxiliary variable is empty, then the test stops and decides that the given `QSNF` is not satisfiable, since there is no possible assignment for that variable. Otherwise, we obtain a non-empty

Input: A QSNF of the form

$$\exists \overline{w} \, (\, \overline{x} \approx \overline{t} \, \wedge \, \bigwedge_{i=1}^{n} [\, \varphi_i^1 \wedge \varphi_i^2 \wedge \psi_i \,] \,)$$

where $\overline{w}$ is an n-tuple of variables, each φ_i^1/ψ_i is a conjunction/disjunction of UQCDs/EQCEs with linear characteristic term and each φ_i^2 is a conjunction of UQCDs with non-linear characteristic term.

Step 1:

```
for each w_i ∈ w̄ do
    Δ_0[w_i] := ChT(w_i, ψ_i)
    Δ_1[w_i] := set_uncover(Δ_0[w_i], φ_i^1)
end do
Δ_1[w̄] := Δ_1[w_1] × ... × Δ_1[w_n]
if Δ_1[w̄] = ∅ then return unsatisfiable
elsif GERVar(Δ_1[w̄], ⋀_{i=1}^n φ_i^2) = ∅ then return satisfiable
end if
```

Step 2:

```
w̄' := GERVar(Δ_1[w̄], ⋀_{i=1}^n φ_i^2)
φ^2 := VarUCDs(⋀_{i=1}^n φ_i^2, w̄')
if set_uncover(Δ_1[w̄'], ChT(w̄', φ^2)) = ∅ then return unsatisfiable
else return satisfiable
end if
```

Fig. 4. A QSNF Satisfiability Test

explicit representation for the auxiliary variables $\overline{w}$ by the Cartesian product of each explicit representation, which have to be refined using the conjunction of UQCDs with a non-linear characteristic term. In the next definition, we provide some conditions that allow to focus this refinement on some (not necessarily all) of the auxiliary variables, also excluding some of the UQCDs from φ_i^2.

Definition 4.7 Let φ be a conjunction of UQCDs with non-linear characteristic term and $\Delta[\overline{w}]$ a non-empty explicit representation for the variables $\overline{w} = \mathtt{Free}(\varphi)$. The *monotonic operator* $V_{\varphi,\Delta[\overline{w}]} : 2^{\overline{w}} \to 2^{\overline{w}}$ is defined by the following rule: given a variable $z \in \overline{w}$ and a subset of variables $\overline{x} \subseteq \overline{w}$, $z \in V_{\varphi,\Delta[\overline{w}]}(\overline{x})$ iff for every term t in the projection $\Delta[z]$ there exists some $\forall \overline{v} \, (\, z \not\approx r \,)$ in φ such that

(i) $\mathtt{Free}(r) \subseteq \overline{x}$

(ii) t and r are unifiable and $\mathtt{mgu}(t, r)$ is a linear substitution

(iii) $\mathtt{mgu}(t,r)_{\restriction \mathtt{Free}(r)}$ is ground

(iv) $\mathtt{range}(\mathtt{mgu}(t,r)) \cap \mathtt{Free}(r) = \emptyset$

Besides, the set of variables $\mathtt{GERVar}(\Delta[\overline{w}], \varphi)$ denotes the greatest subset of $\overline{w}$ that is obtained as the union of all the iterative application of $V_{\varphi,\Delta[\overline{w}]}$ on the empty tuple. □

It is easy to see that, if $\overline{w}' = \mathtt{GERVar}(\Delta[\overline{w}], \varphi)$ and $\overline{w}'' = \overline{w} \setminus \overline{w}'$, then *uncover* invokes *partition* at least once for each term in the projection $\Delta[\overline{w}']$ in order to refine $\Delta[\overline{w}]$ according to φ. Hence, the projection $\Delta[\overline{w}']$ could become empty. But, on the contrary, there exists at least one term t in the projection $\Delta[\overline{w}'']$ that cannot be removed, because t and s are not unifiable or $\theta = \mathtt{mgu}(t,s)$ is non-linear for each $s \in \mathtt{ChT}(\varphi, \overline{w}'')$. Therefore, the projection $\Delta[\overline{w}'']$ cannot become empty and the variables $\overline{w}''$ can be discarded. Obviously, if we can exclude all the variables, then the test stops and returns satisfiable. Otherwise, the test proceeds to the second step with the set of non-discarded variables. Besides, some of the UQCDs with a non-linear characteristic term can also be discarded. In fact, we exclude all the UQCDs where some discarded variable occurs.

Definition 4.8 Let φ be a conjunction of UQCDs such that $\overline{w} = \mathtt{Free}(\varphi)$ and $\overline{z}$ a subset of $\overline{w}$. The conjunction of UQCDs $\mathtt{VarUCDs}(\varphi, \overline{z})$ is given by

$$\{\ \forall \overline{v}\ (\ w \not\approx r\ )\ |\ \forall \overline{v}\ (\ w \not\approx r\ ) \in \varphi \text{ and } (\mathtt{Var}(\ w \not\approx r\ ) \setminus \overline{v}) \subseteq \overline{z}\ \}. \qquad \square$$

In order to discard more variables, we could defined stronger conditions than the ones in Definition 4.7. However, analizing more complicated conditions would become as expensive as the call to *uncover* with all the remaining UQCDs. Hence, we choose the condition in Definition 4.7 because it is not difficult to check and, besides, the set of discarded variables is optimal in most of the cases.

At the second step, the test refines the projection of the explicit representation to the tuple of non-discarded variables using the conjunction of non-excluded UQCDs Finally, our test decides that the input QSNF is satisfiable unless the refinement returns an empty explicit representation.

Next, we illustrate the application of our refined test to some QSNFs in the following two examples.

Example 4.9 Let us consider $\mathcal{F}_{\mathcal{L}} = \{a_{/0},\ g_{/1},\ f_{/2}\}$ and the following QSNF

$$\exists \overline{w}\ (\ x \approx f(w_1, w_2) \wedge w_1 \not\approx a\ \wedge\ \forall v\ (w_1 \not\approx g(v))\ \wedge\ \forall v\ (w_1 \not\approx f(v,v))\ \wedge \tag{20}$$

$$\forall v\ (w_1 \not\approx f(w_2, v))\ \wedge\ [\ \exists z\ (w_1 \approx g(z))\ \vee\ \exists \overline{z}\ (w_1 \approx f(z_1, z_2))\ ] \wedge \tag{21}$$

$$\forall v (w_2 \not\approx g(g(v)))\ \wedge\ \forall \overline{v}\ (w_2 \not\approx f(v_1, v_2))\ \wedge\ w_2 \not\approx g(w_1)\ \wedge \tag{22}$$

$$\forall v\ (w_2 \not\approx f(g(v,v)))\ \wedge\ [\ w_2 \approx a\ \vee\ \exists \overline{z}\ (w_2 \approx f(f(z_1, a), z_2))\ ]\ ) \tag{23}$$

According to the disjunctions of EQCEs in (21) and (23), the initial explicit representations are:

$$\Delta_0[w_1]\ =\ \{g(z),\ f(z_1, z_2)\}$$
$$\Delta_0[w_2]\ =\ \{a,\ f(f(z_1, a), z_2)\}$$

Then, we deal with the conjunctions of UQCDs such that its characteristic term is linear, which are $\varphi_1^1 = w_1 \not\approx a \wedge \forall v\ (\ w_1 \not\approx g(v)\ )$ (the two first UQCDs in (20)) and $\varphi_1^2 = \forall v\ (\ w_2 \not\approx g(g(v))\ ) \wedge \forall \overline{v}\ (\ w_2 \not\approx f(v_1, v_2)\ )$ (the two first UQCDs in (22)). The calls $\mathit{set\_uncover}(\Delta_0[w_1], \mathtt{ChT}(w_1, \varphi_1))$ and $\mathit{set\_uncover}(\Delta_0[w_2], \mathtt{ChT}(w_2, \varphi_2))$ respectively return

$$\begin{aligned} \Delta_1[w_1] &= \{f(z_1, z_2)\} \\ \Delta_1[w_2] &= \{a\} \end{aligned}$$

thus $\Delta[w_1, w_2] = \{\mathtt{c}(f(z_1, z_2), a)\}$ is non-empty. Hence, we have to check if we can discard some of the variables w_1 or w_2 according to the remaining UQCDs: $\varphi_2^1 = \forall v\ (w_1 \not\approx f(v, v)) \wedge \forall v\ (w_1 \not\approx f(w_2, v))$ (the last UQCD in (20) and the one in (21)) and $\varphi_2^2 = w_2 \not\approx g(w_1) \wedge \forall v\ (w_2 \not\approx f(g(v, v)))$ (the last UQCD in (22) and the one in (23)). On one hand, $\mathtt{mgu}(f(z_1, z_2), f(v, v))$ is non-linear and $\mathtt{mgu}(f(z_1, z_2), f(w_2, v))_{\restriction_{w_1 \cdot w_2}}$ is non-ground, hence conditions (2) and (3) in Definition 4.7 are respectively violated. On the other hand, the term $a \in \Delta_1[w_2]$ does not unify with neither $g(w_1)$ or $f(g(v, v))$, thus it violates condition (1) in Definition 4.7. Hence, w_1 and w_2 together with all the remaining UQCDs are discarded and the input QSNF is decided to be satisfiable at the first step. □

Example 4.10 Let us consider $\mathcal{F}_{\mathcal{L}} = \{a_{/0},\ g_{/1},\ f_{/2}\}$ and the following QSNF

$$\exists w\ (\ x \approx f(w, a) \wedge \forall v\ (w \not\approx g(v))\ \wedge\ [\ w \approx g(a) \vee \exists z\ (w \approx g(g(z)))\ ]\ ).$$

The initial explicit representation that is obtained from the disjunction of EQCEs is $\Delta_0[w] = \{g(a),\ g(g(z))\}$ and the call $\mathit{set\_uncover}(\Delta_0[w], \mathtt{ChT}(w, \forall v\ (\ w \not\approx g(v)\ )))$ returns an empty explicit representation, thus the input QSNF is decided to be unsatisfiable at the first step. □

QSNF satisfiability test has a poor worst case performance. Actually, it is an NP-complete problem (see [13]). However, our test performs efficiently in practice because of several structural reasons that can be summed up as follows. In general, QSNFs having expensive computations in both steps are unlikely. If the input QSNF contains a lot of UQCDs to be treated in the first step, then the explicit representation for some variable usually becomes empty and the test stops. However, when few UQCDs are treated at the first step, it is usual that we discard most of the auxiliary variables and, thus, the second step becomes unnecessary or very cheap. On the contrary, the worst case occurs when every $w \in \overline{w}$ has a large explicit representation, but every possible assignment violates some UQCDs. In our experience, the combination of both properties requires a lot of UQCDs to be expressed.

5 Conclusions and Future Work

QSNFs have been proposed as a tool for efficiently solving equality constraints, with the aim of avoiding the repetition of many (and unnecessary) transformations when using the quantifier elimination technique. Roughly speaking, QSNFs provide a more compact representation, which allows to delay many distribution transformations. Besides, we have shown that testing satisfiability on UQCDs is not hard in practice.

Thus, the sequence of conjunction and negation transformations that is the basis a the general equality constraint solver can be more efficiently performed than using less compact normal forms.

Other improvements on general equality constraint solving using QSNFs , which may be studied in future works, are related to the initial transformation of general constraints into prenex formulas whose matrices are a disjunction of QSNFs and, also, to the use of strategies along the quantifier elimination process. Here, we briefly discuss both issues.

On one hand, the pre-process of the initially given constraint is a critical task that dramatically affects the perfomance of the solving process. Two features of this preliminary treatment are of great interest. First, the application of some simplification rules, such as the rules proposed in [6], eliminates superfluous information and serves to reduce the number of QSNFs in the matrix. The second feature is the minimization of the number of variables in the prefix of formulas. That is, the proposed quasi-solved form allows both existential and universal quantifiers. Thus, some quantifiers in the initial formula may be translated into auxiliary quantifiers of a QSNF, instead of being moved to the prefix of the formula. Such a preliminar treatment of formulas could reduce (i) the number of quantifier elimination steps that are necessary for solving general equality constraint, and also (ii) the number of equations/disequations in the matrix of formulas. A combination of both features yields a more compact representation of formulas, which optimizes the solving process.

On the other hand, the cost of resolution grows exponentially with the size of constraints, due to the combination of conjunction and negation. Thus, it is desirable to limitate the maximum size of formulas to be solved. The size of a formula may be given by the number of (a) quantifiers, (b) equations/disequations, (c) variables, or any combination of them. Then, in order to solve formulas that exceed the maximum size, we may apply the classical *Divide and Conquer* technique. That is, if size limitation does not hold, then we first split the formula into subformulas satisfying the maximum size. For solving purposes, if the scope of some variable quantifier $\exists x/\forall x$ is more than one subformula, then the variable x is consider as a free variable in each subformula where x occurs. Then, each subformula is turned into prenex form and its matrix is transformed into a disjunction of QSNFs for the variables in the prefix and the variables that occur free. After solving all these subformulas, we combine the disjunctions of QSNFs obtained by resolution. For the combination phase, it may be necessary to apply the resolution method again if some quantifiers remain in the formula. Locally solving formulas in this way, that is applying the *Divide and Conquer* technique, is several times faster than global resolution.

References

[1] Álvez, J. and P. Lucio, *Equational constraint solving via a restricted form of universal quantification*, in: J. Dix and S. J. Hegner, editors, *Foundations of Information and Knowledge Systems, Proceedings of the 4th International Symposium, FoIKS 2006, Budapest, Hungary, February 14-17, 2006*, Lecture Notes in Computer Science **3861** (2006), pp. 2–21.

[2] Buntine, W. L. and H.-J. Bürckert, *On solving equations and disequations*, J. ACM **41** (1994), pp. 591–629.

[3] Clark, K. L., *Negation as failure*, in: H. Gallaire and J. Minker, editors, *Logic and Data Bases* (1978), pp. 293–322.

[4] Colmerauer, A., *Equations and inequations on finite and infinite trees*, in: *FGCS*, 1984, pp. 85–99.

[5] Comon, H., *Disunification: A survey*, in: J. Lassez and G. Plotkin, editors, *Computational Logic - Essays in Honor of Alan Robinson*, 1991, pp. 322–359.

[6] Comon, H. and P. Lescanne, *Equational problems and disunification*, J. Symb. Comput. **7** (1989), pp. 371–425.

[7] Compton, K. J. and C. W. Henson, *A uniform method for proving lower bounds on the computational complexity of logical theories*, Ann. Pure Appl. Logic **48** (1990), pp. 1–79.

[8] Lassez, J.-L. and K. Marriott, *Explicit representation of terms defined by counter examples*, J. Autom. Reasoning **3** (1987), pp. 301–317.

[9] Lloyd, J. W. and R. W. Topor, *A basis for deductive database systems*, J. Log. Program. **2** (1985), pp. 93–109.

[10] Maher, M. J., *Complete axiomatizations of the algebras of finite, rational and infinite trees*, in: *LICS '1988: Proceedings of the Third Annual Symp. on Logic in Computer Science, 5-8 July 1988, Edinburgh, Scotland, UK* (1988), pp. 348–357.

[11] Maher, M. J., *Equivalences of logic programs*, in: *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 1988 pp. 627–658.

[12] Malcev, A. I., *Axiomatizable classes of locally free algebras*, in: B. F. Wells, editor, *The Metamathematics of Algebraic Systems (Collected Papers: 1936-1967)*, Studies in Logic and the Foundations of Mathematics **66**, North-Holland, 1971 pp. 262–281.

[13] Pichler, R., *On the complexity of equational problems in cnf*, J. Symb. Comput. **36** (2003), pp. 235–269.

[14] Reiter, R., *On closed world data bases*, in: H. Gallaire and J. Minker, editors, *Logic and Data Bases* (1978), pp. 55–76.

[15] Rybina, T. and A. Voronkov, *A decision procedure for term algebras with queues*, ACM Trans. Comput. Log. **2** (2001), pp. 155–181.

[16] Vorobyov, S. G., *An improved lower bound for the elementary theories of trees*, in: M. A. McRobbie and J. K. Slaney, editors, *CADE-13: Proceedings of the 13th International Conference on Automated Deduction*, Lecture Notes in Computer Science **1104** (1996), pp. 275–287.