$ACI1$ constraints*

Agostino Dovier† Carla Piazza‡ Enrico Pontelli§
Gianfranco Rossi¶

Abstract

Disunification is the problem of deciding satisfiability of a system of equations and disequations w.r.t. a given equational theory. In this paper we study the disunification problem in the context of $ACI1$ equational theories. This problem is of great importance, for instance, for the development of complex constraint solvers over *sets*. We start by providing a characterization of the interpretation structures suitable to model the axioms in $ACI1$ theories, and we develop extensions of (general) $ACI1$ which correspond to the "intuitive" Herbrand-based models on the class of conjunctions of equations and disequations. The satisfiability problem is solved using known techniques for the equality constraints and novel methodologies to transform disequation constraints into solved forms. We propose three solved forms, offering an increasingly more precise characterization of the set of solutions. For each solved form we provide the corresponding rewriting algorithm and we study the time complexity for the transformation and the successive satisfiability test. Remarkably, two of the solved forms can be computed and tested in polynomial time. All these solved forms are adequate to be efficiently used in the context of a Constraint Logic Programming system—e.g., they do not introduce arbitrary variable quantifications (as happens in some of the existing solved forms for disunification problems). The novel results achieved open new possibilities in the practical and efficient manipulation of $ACI1$ constraints.
Keywords: *CLP, disunification, equational theories, solved forms, complexity.*

1 Introduction

Equational theories are first-order theories whose axioms are universally quantified equations between first-order terms [26]. A (non-empty) equational theory E forces certain classes of syntactically different terms to be interpreted as the same object in any model of E. For example, if E contains the axiom $X + Y = Y + X$, then the terms $a + b$ and $b + a$ will be interpreted in the same way in any model of E.

However, an equational theory is generally not strong enough to state when two terms *must* be distinguished. As a matter of fact, a structure such as $\underline{\mathbf{1}} = \langle\{\perp\}, (\cdot)^{\underline{\mathbf{1}}}\rangle$, where the interpretation function $(\cdot)^{\underline{\mathbf{1}}}$ maps all terms to the unique element $\perp$, is a model of *any* equational theory. In $\underline{\mathbf{1}}$ any constraint of the form $s \neq t$ is unsatisfiable! If a "wider" structure is chosen, then the satisfiability problem for a set of positive (equations) and negative (disequations) constraints—a.k.a. *disunification problem*—becomes meaningful and complex.

In this paper we tackle this problem in the context of equational theories describing the *associative* (A), *commutative* (C), and *idempotent* (I) nature of one or more function symbols. Constraints in the context of ACI theories (or similar theories for set-like structures) have been shown to be very important from the theoretical as well as the practical point of view [23, 18, 17, 10]. The ultimate goal of our effort is to develop a framework for handling ACI constraints which can be used in *Constraint Logic Programming (CLP).*

The problem of handling positive and negative constraints under equational theories has been explored in the literature. In [11] a general solution to the disunification problem is presented; unfortunately such solution is valid only for *compact* equational theories, and ACI—as discussed later—does not meet this

*Complete proofs of all the results of the paper are available at http://www.cs.nmsu.edu/lldap/prj_lp/papers/dppr99.html
†Dip. Scientifico-Tecnologico, Univ. di Verona, Strada Le Grazie, 37134 Verona (I), dovier@sci.univr.it
‡Dip. di Matematica e Informatica, Univ. di Udine, Via Le Scienze 206, 33100 Udine (I), piazza@dimi.uniud.it
§New Mexico State University, Dept. Computer Science, Box 30001/CS, Las Cruces, NM 88003 (USA), epontell@cs.nmsu.edu
¶Dip. di Matematica, Univ. di Parma, Via M. D'Azeglio 85/A, 43100 Parma (I), gianfr@prmat.math.unipr.it

requirement. The general problem of solving disequations w.r.t. a given equational theory is also addressed in [6]. Here the technique employed to solve the problem is that of transforming disequations into unification problems, whose solution sets—that, for finitary theories can be finitely represented—are exactly the negation of the solution set for the starting problem. The answer to the satisfiability problem is represented as a pair σ, Ψ, called *substitution with exceptions*: θ is a solution of the initial constraints if and only if θ is an instance of the substitution σ and θ is not an instance of any of the substitutions in the set Ψ. This test can be not trivial, as well as the problem of verifying whether σ, Ψ is non-empty. Moreover, substitutions with exceptions correspond to solved form constraints containing universally quantified variables, which makes them unsuitable to be used in the context of a CLP system. More recently, Baader and Schulz [4] developed a general technique capable of combining satisfiability algorithms for disunification in disjoint equational theories. The approach is very general and can be adapted to work for ACI theories on general signatures. However, the method leads to an exponential explosion of alternatives. There seems to be no practical way to obtain "partial" but more efficient solutions from such scheme.

In this paper we present constraint solving techniques to handle equation and disequation constraints under $ACI1$[1] in CLP languages. The presentation starts with a characterization of the structures which are suitable to model the axioms in $ACI1$—the *join-semilattices with bottom*. This is used to explore the issue of satisfiability of positive and negative constraints with respect to the different possible signatures of the language. This analysis captures the relationship between the satisfiability of negative constraints and the "shape" of the interpretation structure.

In the context of a CLP system, uninterpreted function symbols are typically manipulated as finite trees [24]. We develop a first-order theory which extends (general) $ACI1$ and corresponds to $T(\Sigma)/=_{ACI1}$—i.e., the Herbrand Universe modulo the congruence relation imposed by $ACI1$—on the class of conjunctions of positive and negative constraints. This allows us to focus our analysis on the canonical domain of Herbrand terms. In this context we present three solved forms for disunification problems, as well as the algorithms which allow one to transform arbitrary $ACI1$ constraints into any of these three forms. These solved forms meet the general requirements for solved forms constraints—e.g., deciding their satisfiability is trivial and efficient. Furthermore, all these solved forms are adequate to be efficiently used in the context of a CLP system—a property which was missing from some of the solved forms proposed by other researchers [4, 6]. Two of these solved forms (called *implicit* and *intermediate*) can be obtained in polynomial time from any conjunction of disequations. Finally, we show how the results have a direct application to solve set-based constraints, taking advantage of the polynomial nature of the implicit solved form proposed. We also show that the *set-constraints* (see, e.g., [2, 22]) solver contains an $ACI1$ constraint solver over a particular structure for $ACI1$ different from the canonical one.

The results achieved open new possibilities in the *practical* manipulation of $ACI1$ constraints, thus overcoming limitations and inefficiencies present in the existing CLP languages over set structures [18, 14, 23].

The paper is organized as follows. Section 2 provides preliminary definitions and results concerning the models of $ACI1$ and the computation of canonical form terms. We split the problem of $ACI1$ constraint solving in two parts. First, in Section 3 we recall the results for $ACI1$ unification problem—i.e., the manipulation of positive $ACI1$ constraints. Section 4 studies the issue of satisfiability of negative constraints. In Section 5 the different solved forms mentioned before are introduced and analyzed. Finally, Sections 6 and 7 summarize our results and relate them to analogous problems and results presented in the literature.

2 Preliminaries

Throughout the paper we assume the standard notation of first-order logic and constraints [16, 20]. Let us recall some commonly used terms: a first-order *signature* Σ is a set of function and constant symbols. We denote with $ar(f)$ the arity of the function symbol f. $\mathcal{V}$ is a countable set of *variables*. $T(\Sigma, \mathcal{V})$ ($T(\Sigma)$) denotes the set of first-order terms (resp. ground terms) built from Σ and $\mathcal{V}$ (resp. Σ). $T(\Sigma, \mathcal{V})$ is called the *free* Σ*-algebra with generators* $\mathcal{V}$. Given a sequence of terms $t_1, \ldots, t_n$, $vars(t_1, \ldots, t_n)$ denotes the set of all variables which occur in at least one of the terms t_i. Given a term or a formula t, with $|t|$ we denote

[1] $ACI1$ is ACI with an additional axiom requiring the existence of an identity element for the ACI operator.

the number of occurrences of symbols from $\Sigma \cup \mathcal{V}$ in t. We will follow the convention of denoting variables with upper-case characters and function symbols (and constants) with lower-case characters. With $s \equiv t$ we denote the *syntactic equivalence* between the terms s and t.

Given $\mathcal{L} = \langle \Pi, \Sigma, \mathcal{V} \rangle$, where Π is a collection of *predicate symbols*, we denote with Adm a given set of first-order formulae, called *admissible constraints*. Given a first-order theory T on $\mathcal{L}$ and a model $\mathcal{A}$ of T, T and $\mathcal{A}$ *correspond on* Adm [20] if, for each $c \in Adm$, we have that $(T \models \vec{\exists}(c)$ iff $\mathcal{A} \models \vec{\exists}(c)\,)$. A *valuation* of a constraint c is an assignment to the free variables of c of values from A—where A is the domain of $\mathcal{A}$. Given two interpretations $\mathcal{A}$ (over the domain A) and $\mathcal{B}$ (over the domain B) of $\mathcal{L}$, an homomorphism h from $\mathcal{A}$ to $\mathcal{B}$ is a function $h : A \longrightarrow B$ such that: $\forall f \in \Sigma, a_1, \ldots, a_n \in A\ (h(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(h(a_1), \ldots, h(a_n)))$ and $\forall p \in \Pi, a_1, \ldots, a_m \in A\ (p^{\mathcal{A}}(a_1, \ldots, a_m) \rightarrow p^{\mathcal{B}}(h(a_1), \ldots, h(a_m)))$.

2.1 E-unification and E-disunification

If $s, t \in T(\Sigma, \mathcal{V})$, then $s = t$ is a Σ-*equation* and $s \neq t$ is a Σ-*disequation*. An *equational theory* is a first-order theory whose axioms are universally quantified Σ-equations.

Given an equational theory E, we can define the concept of E-*equality* ($=_E$) as the *least congruence relation* over $T(\Sigma, \mathcal{V})$ which contains E and which is closed under substitution [5]. The relation $=_E$ induces a partition of $T(\Sigma)$ into congruence classes. The set of these classes will be denoted by $T(\Sigma)/=_E$. $T(\Sigma)/=_E$, together with a mapping (the interpretation function) which assigns to each term t the the equivalence class $[t]$, is a model of the theory E.

Given a conjunction $C \equiv (s_1 = t_1 \wedge \ldots \wedge s_n = t_n)$ of Σ-equations, the *(decision) E-unification problem* is the problem of deciding whether $E \models \vec{\exists} C$. If E is an equational theory, Birkoff's completeness theorem [26] ensures that $E \models \vec{\exists} C$ if and only if $T(\Sigma)/=_E \models \vec{\exists} C$. From a constraint point of view, Σ-equations can be chosen as admissible constraints. The theory E and the model identified by $T(\Sigma)/=_E$ correspond on the class containing all possible conjunctions of equations [20].

A substitution σ is an E-*unifier* of two terms s, t if $s^\sigma =_E t^\sigma$—i.e., s^σ and t^σ belong to the same E-congruence class [26]. A collection of E-unifiers for s, t is *complete* if every E-unifier of s, t can be expressed as instance of a unifier in the collection. The elements of a complete and minimal set of unifiers are usually called *most general unifiers*. The *E-unification problem* of s and t is the problem of finding a (minimal) set of unifiers which represents all solutions to the equation $s = t$.

Given a conjunction $C \equiv (s_1 \neq t_1 \wedge \ldots \wedge s_n \neq t_n)$ of Σ-disequations (i.e., a disequation constraint), we call *(decision) E-disequation problem* the problem of establishing whether $E \models \vec{\exists} C$. For a equational theory E, this test has always a negative answer, since, for instance, the structure $\underline{\mathbf{1}} = \langle \{\perp\}, (\cdot)^{\underline{\mathbf{1}}} \rangle$, with $(\cdot)^{\underline{\mathbf{1}}}$ the interpretation of all terms in the unique element $\perp$, is a model of any equational theory, and in $\underline{\mathbf{1}}$ any constraint of the form $s \neq t$ is unsatisfiable. This problem originates from the fact that any (non-empty) equational theory E forces certain distinct terms to be interpreted in the same way in any model of E—however, it is not strong enough to state when two terms *must* be distinguished in each model of E. As a consequence, the disequation problem is typically stated as the problem of verifying satisfiability of $\vec{\exists} C$ w.r.t. a given interpretation structure $\mathcal{A}$ (usually, $T(\Sigma)/=_E$)—i.e., $\mathcal{A} \models \vec{\exists} C$. A related problem is the determination of the structures $\mathcal{A}$ which fulfill such property.

Example 2.1 *Let E consist of the unique axiom $X = Y$ and let* Adm *contain all Σ-equations and Σ-disequations. Then E corresponds with $\underline{\mathbf{1}}$ (in particular, E is the complete theory of $\underline{\mathbf{1}}$).*

If E corresponds with $\underline{\mathbf{1}}$, then E is said to be *trivial*. In particular:

Proposition 2.2 *Given a non-trivial equational theory E such that the set* Adm *contains all Σ-equations and Σ-disequations, there is no corresponding structure for E.*

Proof. Assume, by contradiction, that there exists a structure $\mathcal{A}$ which corresponds to E. Since E is not trivial, $\mathcal{A} \neq \underline{\mathbf{1}}$. Thus, $|A| \geq 2$ and there are $a, b \in A$ such that $a \neq b$. This implies that $\mathcal{A} \models \exists X, Y (X \neq Y)$; but $E \not\models \exists X, Y (X \neq Y)$ since $\underline{\mathbf{1}}$ is a model of E and $\underline{\mathbf{1}} \not\models \exists X, Y (X \neq Y)$. □

A theory E is *satisfaction complete* [20] if for each admissible constraint c either $E \models \vec{\exists} c$ or $E \models \neg\vec{\exists} c$. In terms of satisfaction completeness of non-trivial theories, Proposition 2.2 leads to the following result:

Corollary 2.3 *Given a non-trivial equational theory E such that the set* Adm *contains all Σ-equations and Σ-disequations, E is not satisfaction complete.*

As mentioned in Sect. 1, our ultimate goal is to handle constraints composed by arbitrary conjunctions C of Σ-equations and Σ-disequations. This class of problems has been typically referred to as *E-disunification problems* [11, 6].[2] An *E-solution* σ of C in a structure $\mathcal{A}$—denoted $\mathcal{A} \models C\sigma$—is a valuation $\sigma : \mathcal{V} \longrightarrow A$, extended as usual to terms, such that $s\sigma =_{\mathcal{A}} t\sigma$ for all $s = t$ in C and $s\sigma \neq_{\mathcal{A}} t\sigma$ for all $s \neq t$ in C.

When the domain of $\mathcal{A}$ is $T(\Sigma)/=_E$ ($T(\Sigma, \mathcal{V})/=_E$), if the constraint C admits a solution, then it is said to be *ground solvable* (*non-ground solvable*). In the case of $ACI1$, and in presence of a signature containing non-constant free function symbols, ground and non-ground solvability of disunification are equivalent problems [4].

The technique for handling equations and disequations presented in this paper provides us with a methodology to tackle disunification problems. In fact, if E is a finitary theory—i.e., every unification problem admits a finite complete collection of most general unifiers—then a disunification problem C can be simply solved by computing a complete set of unifiers for the equations in C and then verifying whether, given any of these unifiers σ, there is a solution for the disequations of C^σ [4]. Clearly, here we are taking advantage of the fact that unification problems can be not only decided, but a collection of unifiers for such problems can be explicitly computed.

2.2 $ACI1$ and its Models

Let $\Sigma \supseteq \{\emptyset, \cup\}$ be a signature containing the binary function symbol $\cup$ and the constant symbol $\emptyset$, and let $\mathcal{V}$ be a countable set of variables. Let us also define a one-to-one function $\# : \Sigma \cup \mathcal{V} \longrightarrow \mathbb{N}$, which will be used to obtain an order over $T(\Sigma, \mathcal{V})$. Observe that $\emptyset$ and $\cup$ can be replaced by any other pair of function symbols fulfilling the same axioms. A signature Σ is *general* if it contains at least one function symbol different from $\cup$ of arity greater than 0.

Let us also recall some standard definitions from lattice theory [19]. A relation $\leq \subseteq L \times L$ is a *partial order* on L if $\leq$ is reflexive, antisymmetric, and transitive; let us denote with $\perp$ the *bottom* of the partial order—i.e., $(\forall x \in L)(\perp \leq x)$. $\langle L, \leq \rangle$ is a *join-semilattice* if the element $x \bigvee y \in L$ exists for each $x, y \in L$, where $x \bigvee y$ (read "x join y") is the unique element satisfying $x \leq x \bigvee y, y \leq x \bigvee y$ and for all $z \in L$ such that $x \leq z$ and $y \leq z$ it holds that $x \bigvee y \leq z$. If $\langle L, \leq \rangle$ is a partial order with bottom $\perp$, then $a \in L$ is an *atom* if $(\forall x \in L)((x \leq a \wedge x \neq a) \to x = \perp)$.

The following equations describe the theory $ACI1$:

$$\begin{array}{llcl@{\qquad\qquad}llcl} (A) & (X \cup Y) \cup Z & = & X \cup (Y \cup Z) & (I) & X \cup X & = & X \\ (C) & X \cup Y & = & Y \cup X & (1) & \emptyset \cup X & = & X \end{array}$$

Let us analyze the structures $\mathcal{A} = \langle A, (\cdot)^{\mathcal{A}} \rangle$ for $\Sigma = \{\emptyset, \cup\}$ that are models of $ACI1$. By definition of structure, the domain A is not empty. We indicate $(\emptyset)^{\mathcal{A}} \in A$ as $\perp$. A relation $\leq$ is induced by $(\cup)^{\mathcal{A}}$ (simply denoted by $\cup^{\mathcal{A}}$) on A: $x \leq y \leftrightarrow x \cup^{\mathcal{A}} y = y$.

Proposition 2.4 *Let $\mathcal{A} = \langle A, (\cdot)^{\mathcal{A}} \rangle$ be a model of ACI. Then $\leq$ is a partial order on A. Moreover, if $\mathcal{A}$ is a model of the axiom* (1)*, then $\perp = (\emptyset)^{\mathcal{A}}$ is the bottom of A.*

Proof. $\leq$ is reflexive, since, by (I) $x \cup^{\mathcal{A}} x = x$ for all $x \in A$. $\leq$ is antisymmetric, since, by (C), $x \cup^{\mathcal{A}} y = y$ and $y \cup^{\mathcal{A}} x = x$ implies $x = y$.

Assume $x \cup^{\mathcal{A}} y = y$ and $y \cup^{\mathcal{A}} z = z$. Then $x \cup^{\mathcal{A}} z = x \cup^{\mathcal{A}} (y \cup^{\mathcal{A}} z)$. By (A) the latter is equal to $(x \cup^{\mathcal{A}} y) \cup^{\mathcal{A}} z = y \cup^{\mathcal{A}} z = z$, which proves the transitivity of $\leq$. Finally, by (1) $(\forall x \in A)(((\emptyset)^{\mathcal{A}} \cup^{\mathcal{A}} x = x)$, i.e., $\perp \leq x$. □

[2] Observe that this definition of *disunification* is different then the one we used before.

```
function ρ(t):
  if t is of the form f(t_1,...,t_h), h ≥ 0, f ≢ ∪
  then return f(ρ(t_1),...,ρ(t_h))
  else begin
    t_1 ∪ ··· ∪ t_h := η(t);
    for i := 1 to h do t_i := ρ(t_i);
    L := top([t_1,...,t_h]);
    [s_1,...,s_m] := remove repetitions from L;
    if m > 1 and s_1 = ∅
      then return s_2 ∪ ··· ∪ s_m
      else return s_1 ∪ ··· ∪ s_m
  end;

function η(t):
  if t is of the form (t_1 ∪ t_2)
  then return η(t_1) ∪ η(t_2)
  else return t;
```

Figure 1: Normalization function

It is easy to observe that if $\mathcal{A}$ is not a model of ACI, then $\leq$ is not guaranteed to be a partial order. Given a structure $\mathcal{A}$ for $\{\emptyset, \cup\}$ that is a model of $ACI1$, we denote with $\langle A, \leq \rangle$ the partial order defined above [19].

Proposition 2.5 *Let $\mathcal{A} = \langle A, (\cdot)^{\mathcal{A}} \rangle$ be a structure for $\{\emptyset, \cup\}$ with $\perp = (\emptyset)^{\mathcal{A}} \in A$. $\mathcal{A}$ is a model of $ACI1$ if and only if $\langle A, \leq \rangle$ is a join-semilattice with bottom $\perp$.*

Proof. Let $\mathcal{A}$ be a model of $ACI1$. Prop. 2.4 ensures that $\leq$ is a partial order on A with bottom $\perp$. Hence $\langle A, \leq \rangle$ is a join-semilattice when $x \bigvee y$ is taken as $x \cup^{\mathcal{A}} y$.

Let $\langle A, \leq \rangle$ be a join-semilattice with bottom $\perp$. Now $\mathcal{A} = \langle A, (\cdot)^{\mathcal{A}} \rangle$ is defined as $\perp = (\emptyset)^{\mathcal{A}} \in A$ and $\forall a_1, a_2 \in A\ (a_1 \cup^{\mathcal{A}} a_2 = a_1 \bigvee a_2)$. Observe that $a_1 \bigvee a_2$ exists since $\langle A, \leq \rangle$ is a join-semilattice. We prove that $\mathcal{A}$ is a model of $ACI1$ (we use standard properties of join and the definition of bottom):

$$\begin{array}{llrcl}
(A) & \forall a_1, a_2, a_3 \in A & (a_1 \cup^{\mathcal{A}} a_2) \cup^{\mathcal{A}} a_3 & = & (a_1 \bigvee a_2) \bigvee a_3 = a_1 \bigvee (a_2 \bigvee a_3) \\
 & & & = & a_1 \cup^{\mathcal{A}} (a_2 \cup^{\mathcal{A}} a_3) \\
(C) & \forall a_1, a_2 \in A & a_1 \cup^{\mathcal{A}} a_2 & = & a_1 \bigvee a_2 = a_2 \bigvee a_1 = a_2 \cup^{\mathcal{A}} a_1 \\
(I) & \forall a_1 \in A & a_1 \cup^{\mathcal{A}} a_1 & = & a_1 \bigvee a_1 = a_1 \\
(1) & \forall a_1 \in A & \emptyset^{\mathcal{A}} \cup^{\mathcal{A}} a_1 & = & \perp \bigvee a_1 = a_1
\end{array}$$

□

2.3 Term Normalization

The $ACI1$ axioms allow us to design a normalization function for $T(\Sigma, \mathcal{V})$-terms, $\rho : T(\Sigma, \mathcal{V}) \longrightarrow T(\Sigma, \mathcal{V})$, as described in Fig. 1. Intuitively, the effect of ρ is to remove repeated elements and occurrences of $\emptyset$ from unions and reorder the elements of a union according to the ordering induced by $\#$. For example, given a term $t \in T(\{\emptyset, \cup\}, \mathcal{V})$, then $\rho(t)$ is always of the form: $\emptyset$ or $X_1 \cup \ldots \cup X_m$ where $\#(X_1) < \cdots < \#(X_m)$. Observe that the result of $\rho(t)$ is not properly a term, but the associativity of $\cup$ allows us to use this entities as terms. top is a lexicographical sorting algorithm that accepts as input lists of terms from $T(\Sigma, \mathcal{V})$, and uses $\#$ as the ordering criterion to compare basic components of the terms. $\eta(t)$ is an auxiliary function that removes all the outermost parentheses when the term t is of the form $((\cdots(t_1 \cup t_2) \cup \cdots) \cup t_h)$.

Theorem 2.6 *Let $S = \Sigma \setminus \{\emptyset, \cup\}$. If $t \in T(\Sigma, \mathcal{V})$, $vars(t) = \bar{X}$, $|t| = n$, then $ACI1 \models \forall \bar{X}\, (\rho(t) = t)$ and*

1. *if S is a set of constant symbols, then $\rho(t)$ can be performed in time $O(n \log n)$,*
2. *if S contains a function symbol of arity greater than 0, then $\rho(t)$ can be performed in time $O(n^2)$.*

Proof. The first part is an immediate consequences of the axioms $ACI1$.

1. In the first case, there are no occurrences of function symbols in t. Thus, there is only one list to sort and it is made by strings of length 1. The time complexity is $O(n)$ for η, $O(n \log n)$ for the ordering and $O(n)$ to remove repetitions; the last instruction has cost $O(1)$.

2. From [1] (Theorem 3.2, pag. 83) we know that top has complexity $O(\ell_{tot} + m)$, where m is the cardinality of the alphabet and ℓ_{tot} is the sum of the lengths of the strings to order. We can consider the alphabet composed by the set of function and constant symbols and variables occurring in t. The extraction of the alphabet can be made in time $O(n \log n)$ before calling ρ (for the sake of simplicity, we do not count parentheses and commas). Thus, $m \leq n$ and $\ell_{tot} \leq n$: the complexity of top is $O(n)$.

We compute the time complexity $T(n)$:

$$\begin{cases} T(1) & = & c & \text{when } t \text{ is a constant or a variable term} \\ T(n) & = & \sum_{i=1}^{h} T(\ell_i) + c & \text{when } t \text{ is } f(t_1, \ldots, t_h),\ h > 0, \text{ and } \ell_i \text{ is the length of } t_i \\ T(n) & = & \sum_{i=1}^{h} T(\ell_i) + O(n) & \text{when } \eta(t) \text{ is } t_1 \cup \cdots \cup t_h,\ h > 0, \text{ and } \ell_i \text{ is the length of } t_i \end{cases}$$

The worst case is the last one. We prove, by induction that $T(n) = O(n^2)$.

Base: For $n = 1$ the result is trivial.

Step: Assume that $T(\ell) = O(\ell^2) \leq k\ell^2$, for some $k \in \mathbb{N}$, for all $\ell < n$. Since $(*)$ $\sum_{i=1}^{h} \ell_i \leq n - 1$, then we have:

$$\begin{array}{rcll} T(n) & = & \sum_{i=1}^{h} T(\ell_i) + O(n) & \text{by definition} \\ & \leq & k \sum_{i=1}^{h} (\ell_i)^2 + O(n) & \text{by inductive hypothesis} \\ & \leq & k(\sum_{i=1}^{h} \ell_i)^2 + O(n) & \text{by algebraic properties} \\ & \leq & k(n-1)^2 + O(n) & \text{by } (*) \end{array}$$

Since we can assume k big enough to guarantee that $k(n-1)^2 + O(n) \leq kn^2$ and we have proved the bound using the same constant k of the inductive hypothesis, we have that $T(n) = O(n^2)$. □

Given a term t of $T(\Sigma)$ we will say that t is *normalized* if $t \equiv \rho(t)$. We have that $\rho(t)$ can be chosen as the canonical representative of the $ACI1$-congruence class $[t]$ in $T(\Sigma)$ or $T(\Sigma, \mathcal{V})$ to which t belongs.

Given a conjunction $C \equiv (s_1 \pi_1 t_1 \wedge \ldots \wedge s_h \pi_h t_h)$ where $\pi_i \in \{=, \neq\}$, its *canonical form* is the formula $\rho(C) \equiv \rho(s_1)\pi_1\rho(t_1) \wedge \ldots \wedge \rho(s_h)\pi_h\rho(t_h)$. The worst-case time complexity for the computation of the canonical form of C is $O(n^2)$, where $n = |C|$. In particular, ρ is an idempotent operation, thus $\rho(t) \equiv \rho(\rho(t))$ for all terms t.

Corollary 2.7 *Let $t_1, t_2 \in T(\Sigma, \mathcal{V})$. Then $ACI1 \models \vec{\forall}(t_1 = t_2)$ if and only if $\rho(t_1) \equiv \rho(t_2)$.*

Proof. It derives from Prop. 2.6 and from the uniqueness of the results of the lexicographical ordering, without repetitions. □

Corollary 2.8 *If $C \equiv (s_1 \pi_1 t_1 \wedge \ldots \wedge s_h \pi_h t_h)$, where $\pi_i \in \{=, \neq\}$, then $ACI1 \models \vec{\forall}(C \leftrightarrow \rho(C))$.*

Proof. It follows from Corollary 2.7.

3 $ACI1$ Equation Constraints - Unification

Given two terms s, t we are interested in the decision problem $ACI1 \models \vec{\exists}(s = t)$ and in computing a complete set of $ACI1$ unifiers. Thanks to Corollary 2.8, we can concentrate on the problem $\rho(s) = \rho(t)$. The problem can be classified in three possible classes, according to the form of the signature Σ:

Elementary Unification: Σ contains exclusively the two symbols $\cup$ and $\emptyset$; i.e., $\Sigma = \{\emptyset, \cup\}$. The *decision* problem in this case admits always an affirmative answer—the valuation $[V/\emptyset \ : \ V \in vars(s,t)]$ is always a solution. $\rho(s) = \rho(t)$ is of the form:[3]

$$X_1 \cup \ldots \cup X_m = Y_1 \cup \ldots \cup Y_n$$

If $m = 0$ and $n = 0$, the unique most general unifier is the empty substitution ε. When $m > 0$ and $n = 0$, the unification has the unique solution: $[X_1/\emptyset, \ldots, X_m/\emptyset]$ (similarly when $m = 0$ and $n > 0$). The case $m > 0, n > 0$ is more complex. However, a unique most general solution exists and it can be easily computed using the technique of [3].

Unification with Constants: Σ contains the two symbols $\cup$ and $\emptyset$, together with a finite collection of constants $\{c_1, \ldots, c_n\}$ distinct from $\emptyset$; i.e., $\Sigma = \{\cup, \emptyset, c_1, \ldots, c_n\}$. $\rho(s) = \rho(t)$ is of the form:

$$X_1 \cup \ldots \cup X_k \cup b_1 \cup \ldots \cup b_h = Y_1 \cup \ldots \cup Y_q \cup d_1 \cup \ldots \cup d_p$$

The decision problem $ACI1 \models \vec{\exists}(\rho(s) = \rho(t))$ can be solved in time $O(n)$ where $n = |\rho(s)| + |\rho(t)|$. First of all we can observe that there are structures for $ACI1$ in which all the constant symbols are mapped to different domain objects and such that the union of different constants is mapped to different objects. Thus, there are equations that are unsatisfiable in some models (e.g., the equation $\emptyset = c_1$ is unsatisfiable in any structure which maps c_1 to an object different from $\perp$). A simple (linear-time, thanks to the ordering) test of inclusion between the constant symbols of the two terms is sufficient to verify satisfiability of the equation.

In [3] it is shown how to compute the (minimal) complete set of most general unifiers for this problem. Uniqueness of the most general unifier is lost, due to the presence of constants, but the problem remains *finitary*—i.e., it is possible to describe the complete set of solutions through a finite number of unifiers.

General Unification: Σ contains an arbitrary collection of symbols without restrictions on their number and arity; i.e., $\Sigma = \{\cup, \emptyset, f_1, f_2, \ldots\}$ (a general signature)

The general unification problem has the following format:

$$X_1 \cup \ldots \cup X_h \cup s_1 \cup \ldots \cup s_k = Y_1 \cup \ldots \cup Y_p \cup t_1 \cup \ldots \cup t_q$$

where s_i, t_j are terms whose main functor is different from $\cup$.

The decision problem is NP-complete [21, 13]. Algorithms to compute complete collections of unifiers for this class of problems have been presented in the literature—either as combination of simpler unification procedures [5] or as ad-hoc unification algorithms [15].

4 $ACI1$ Disequation Constraints - Disunification

In this section we will concentrate on the problem of handling conjunctions of disequations. The whole disunification problem (conjunctions of equations and disequations) can be solved in two stages. First we use the unification techniques of the previous section on the equations of the constraint. Then, after applying the resulting substitutions, we can safely remove all equations and concentrate on the problems $\rho(s) \neq \rho(t)$. While in the unification case each equation is satisfiable in at least one model of $ACI1$, the same does not hold in the case of disequations: a negative constraint can be unsatisfiable in all models (e.g., $\emptyset \neq \emptyset$). Other constraints (e.g., $X_1 \neq X_2$) are satisfiable in some structures and unsatisfiable in others. Thus, the study of $ACI1$ disequations requires an in-depth analysis of the possible structures for the considered theory.

[3] When $m = 0$ the l.h.s. is simply $\emptyset$ and, similarly, if $n = 0$ then the r.h.s. is $\emptyset$.

4.1 Elementary $ACI1$ Disequation Constraints

Let us consider $\Sigma = \{\emptyset, \cup\}$. Given a term t, $\rho(t)$ can be either $\emptyset$ or $X_1 \cup \ldots \cup X_m$. Thus, a disequation in canonical form is described by one of the following cases:

1. $r \neq r$ (i.e., $\emptyset \neq \emptyset$ or $X_1 \cup \cdots \cup X_m \neq X_1 \cup \cdots \cup X_m$)
2. $X_1 \cup \cdots \cup X_m \neq \emptyset$, $m > 0$
3. $X_1 \cup \cdots \cup X_i \cup \cdots \cup X_m \neq Y_1 \cup \cdots \cup Y_n$, $m, n > 0$, such that there exists an X_i which does not occur among $Y_1, \ldots, Y_n$

Cases 2 or 3 can be eventually obtained by swapping the left-hand side and right-hand side of $\rho(s) \neq \rho(t)$. The first disequation is clearly false in any model of E. A disequation of the form 2 or 3 is false in $\underline{\mathbf{1}}$ but it is satisfiable in any other structure for $ACI1$. To obtain a successful valuation σ for form 2 and 3 we can simply fix an element of the domain distinct from $\perp$, say a, and put $\sigma(X_1) = a$, and $\sigma(Z) = \perp$ for all variables Z different from X_1. As far as a disequation constraint of the form $C \equiv (s_1 \neq t_1 \wedge \ldots \wedge s_n \neq t_n)$ is concerned, the following theorem gives a sufficient condition for its satisfiability.

Theorem 4.1 *Let C be a disequation constraint in canonical form containing N primitive constraints of the 3rd kind and K primitive constraints of the 2nd kind. Let $M = N$ if $K = 0$ and $M = N + 1$ if $K > 0$. C is satisfiable in any structure containing a substructure isomorphical to $\langle \wp(\{a_1, \ldots, a_M\}), \subseteq \rangle$.*

Proof. (Sketch) Let X_{δ_i} $i = 1, \ldots, N$ be a variable occurring in the left-hand side but not in the right-hand side of the ith primitive constraint. A solution σ can be built by assigning $\{a_i\}$[4] to X_{δ_i} if X_{δ_i} has not been assigned yet. If there are also equations of type 2, then $\{a_{N+1}\}$ is assigned to all other variables; otherwise, all the remaining variables are assigned $\perp$. □

Corollary 4.2 *If the structure is wide enough—i.e., it contains a substructure isomorphical to a sufficiently large boolean lattice—then the satisfiability problem for a disequation constraint C can be decided with time complexity $O(n \log n)$, where $n = |C|$.*

If, on the other hand, the structure $\mathcal{A}$ is fixed a priori (and possibly not wide enough—according to Theorem 4.1), the satisfiability problem becomes NP-complete:

Theorem 4.3 *Let $\mathcal{A} = \langle A, \leq \rangle$ be a join-semilattice with bottom containing k elements. Deciding the satisfiability of an $ACI1$-constraint in $\mathcal{A}$ is NP-complete.*

Proof. Given the structure $\mathcal{A}$, with its interpretation of $\cup$ and an A-valuation σ of the variables of the constraint C, it is easy to to infer if σ is a solution or not.

To prove hardness, consider the following polynomial reduction. Let $G = \langle V, E \rangle$ be a graph. The k-coloring problem for G is NP-complete [12]. Consider the system of disequations

$$I = \{X \neq Y : X, Y \in V, \{X, Y\} \in E\}$$

The satisfiability of the k-coloring problem for G is equivalent to the satisfiability of I over $\mathcal{A}$. □

The NP-hard nature of the problem justifies, in presence of a fixed domain, a preliminary polynomial test to check if the domain is "wide enough".

[4] $\{a_i\}$ are the *atoms* of the lattice.

4.2 $ACI1$ Disequation Constraints with Constants

Let $\Sigma = \{\emptyset \equiv c_0, \cup, c_1, \ldots, c_m\}$. Let us analyze the structures over such Σ. All models $\mathcal{A}$ for elementary $ACI1$ (thus, all join-semilattices with bottom—Prop. 2.5) are also models of $ACI1$ on Σ, provided an interpretation for the constant symbols over A is given. However, it is natural to focus on structures in which the m constants are interpreted as distinct objects, each of them different from $\perp$. This is equivalent to the introduction of an additional axiom in the theory $ACI1$:

$$(F_2') \quad c_i \neq c_j \quad i, j \in \{0, \ldots, m\}, i \neq j$$

Such structures are exactly all the join-semilattices with bottom with a domain of at least $m + 1$ objects. (F_2') is actually an instance of the *freeness axiom scheme* (F_2) of *Clark's Equational Theory* [9]. For example, if $m = 4$, all the structures of Figure 2 are models of such extended theory.

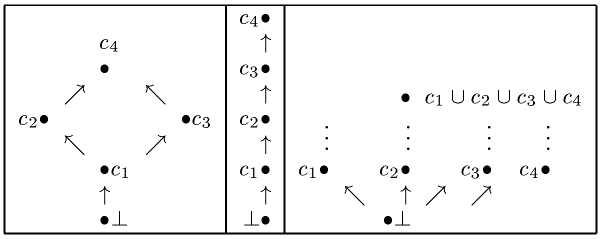

Figure 2: Some models of $ACI1$ when $\Sigma = \{\emptyset, \cup, c_1, c_2, c_3, c_4\}$

Among the possible models, we are interested in those fulfilling the additional axiom

$$(D_c) \quad (\forall I, J \subseteq \{1, \ldots, m\}) \left(I \neq J \rightarrow \bigcup_{i \in I} c_i \neq \bigcup_{j \in J} c_j \right)$$

(D stands for Domain, c for constants), where, if $A = \{a_1, \ldots, a_n\} \subseteq \{1, \ldots, m\}$, then $\bigcup_{i \in A} c_i$ represents the term $c_{a_1} \cup \cdots \cup c_{a_n}$. For instance, when $m = 2$, (D_c) becomes:

$$\emptyset \neq c_1 \wedge \emptyset \neq c_2 \wedge \emptyset \neq c_1 \cup c_2 \wedge c_1 \neq c_2 \wedge c_1 \neq c_1 \cup c_2 \wedge c_2 \neq c_1 \cup c_2$$

Among the structures satisfying these requirements, we can find the Boolean lattices, isomorphical to $\langle \wp(\{c_1, \ldots, c_m\}), \subseteq \rangle$, i.e., those having $\{c_1\}, \ldots, \{c_m\}$ as atoms. Assuming (D_c) we can also ignore (F_2'), since (D_c) implies (F_2').

Let us assume that $\#(\emptyset) = 0$ and $\#(c_i) = i$ for $i = 1, \ldots, m$; using ρ we can focus on the terms of the form $\emptyset$ or

$$X_1 \cup \cdots \cup X_k \cup c_{i_1} \cup \cdots \cup c_{i_h}$$

where $h + k > 0$ and $i_j < i_{j+1}$ for $j = 1, \ldots, h - 1$. The disequation $\rho(s) \neq \rho(t)$ gives rise to the following possible cases:

1. $r \neq r$
2. $c_{i_1} \cup \cdots \cup c_{i_h} \neq c_{j_1} \cup \cdots \cup c_{j_k}$ and $\{i_1, \ldots, i_h\} \neq \{j_1, \ldots, j_k\}$
3. $X_1 \cup \cdots \cup X_m \cup c_{i_1} \cup \cdots \cup c_{i_h} \neq Y_1 \cup \cdots \cup Y_n \cup c_{j_1} \cup \cdots \cup c_{j_k}$, $m > 0$.

Disequations of the 1st form are false in any model of $ACI1$. Disequations of 2nd kind are true (and therefore can be removed) in any model of $ACI1 + (D_c)$. In particular, given any join-semilattice with bottom different from $\underline{\mathbf{1}}$, it is possible to build a structure which is a model of $ACI1$ and in which a given disequation of type 2 is satisfied. Similarly, satisfiability of disequations of the 3rd kind depends on the domain. In particular, a disequation of type 3 is:

- unsatisfiable in $\underline{\mathbf{1}}$
- satisfiable in any model of $ACI1D_c$ if there is a constant which occurs on one side and not on the other, or if there is a constant in Σ which does not occur in the disequation.

The following theorem holds independently from the presence of (D_c):

Theorem 4.4 *If C is a disequation constraint in canonical form and it contains r disequations (of type 2 or 3), then C is satisfiable in any structure which contains a substructure isomorphical to $\langle \wp(\{a_1, \ldots, a_r\}), \subseteq \rangle$.*

Proof. (Sketch) Let us assume that the disequations are numbered. The ith disequation contains an element (variable or constant) ν_i which appears on the left-hand side and does not appear on the right-hand side—otherwise it would be an unsatisfiable disequation of type 1. Let us construct an interpretation $(\cdot)^{\mathcal{A}}$ and a valuation σ as follows: if ν_i is a constant which has not been interpreted yet, then $\nu_i^{\mathcal{A}} = \{a_i\}$, otherwise if ν_i is a variable which has not been assigned yet, then $\sigma(\nu_i) = \{a_i\}$. $\square$

The proof of the following corollary is a simple generalization of Corollary 4.2 and Theorem 4.3.

Corollary 4.5 *If the structure is wide enough, then the decision problem for a disequation constraint C can be solved in $O(n \log n)$, where $n = |C|$. For a given fixed structure, the problem is NP-complete.*

4.3 General $ACI1$ Disequation Constraints

Let us assume that the signature Σ can contain any constant and function symbols. From Section 2.2 we know that when $\leq$ is induced by the interpretation of $\cup$, the models of $ACI1$ are the join-semilattices with bottom. However, in presence of a domain A and an interpretation $\cup^{\mathcal{A}}$, the interpretation of the function symbols of Σ introduces a variety of possibilities for building models.

4.3.1 Characterization of a Privileged Model

The most common interpretation of the function symbols different from $\cup$, including the constant symbols, is the one induced by the structure $T(\Sigma)/=_{ACI1}$, denoted by $\mathcal{H}$. In this section we prove some simple results about this model and we define a theory T such that $\mathcal{H}$ and T correspond on the class of formulae we are interested in, namely conjunctions of equations and disequations.

Example 4.6 *Figure 3 shows a representation of $\mathcal{H} = (T(\Sigma)/=_{ACI1})$ when $\Sigma = \{\emptyset, \cup, \{\cdot\}\}$ where $\{\cdot\}$ is a function symbol of arity 1. With a slight abuse of notation, we denote with $\{s,t\}$ the congruence class of $\{s\} \cup \{t\}$. This allows one to interpret $\mathcal{H}$ as the set of* hereditarily finite and well-founded *sets. Sets at level i contain exactly i elements.*

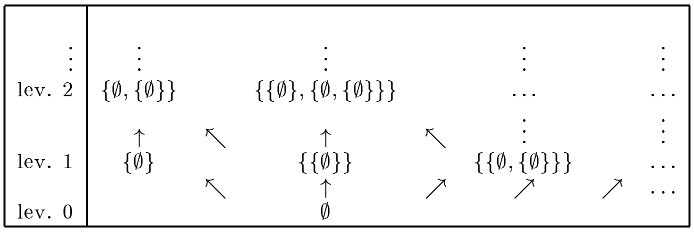

Figure 3: $T(\{\emptyset, \cup, \{\cdot\}\})/=_{ACI1}$

Let us start by observing that the atoms of $\langle \mathcal{H}, \leq \rangle$ are all and only the congruence classes containing terms with a main functor different from $\cup$. This is stated by the following lemma:

Lemma 4.7 *The atoms of $\langle \mathcal{H}, \leq \rangle$ are exactly the classes $[t]$, for some ground term $t \equiv f(t_1, \ldots, t_n)$, $f \not\equiv \cup$.*

Proof. We first prove that classes induced by terms of that form are atoms. Assume $[X] \leq [t]$. Then, $f(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) \cup X$. This equation has only two solutions in $\mathcal{H}$: $X = \emptyset$ and $X = f(t_1, \ldots, t_n)$.

Consider now a representative class of the union of (different) terms, for instance $r = [f(s_1, \ldots, s_m) \cup g(t_1, \ldots, t_n)]$, and assume $ACI1 \not\models f(s_1, \ldots, s_m) = g(t_1, \ldots, t_n)$. Thus, $[f(s_1, \ldots, s_m)] \neq r$. Moreover, $[f(s_1, \ldots, s_m)] \leq r$ and $[f(s_1, \ldots, s_m)] \neq \emptyset$. By definition, r is not an atom. $\square$

Thus, with slight abuse of notation, from now on we will call *atom* any term whose main (outermost) function symbol is different from $\cup$. Observe that this property is not guaranteed to hold in other structures.

For example, in any structure $\mathcal{A}$ in which $\mathcal{A} \models f(t_1, \ldots, t_n) \cup a = f(t_1, \ldots, t_n)$, for some $a \in \Sigma$, the term $f(t_1, \ldots, t_n)$ is not an atom. $\mathcal{H}$ also properly models the extensionality principle for equality between sets denoted by ACI terms:

Lemma 4.8 *Let s, t be two terms and $\mathcal{H} \models \vec{\exists}(s \neq t)$. For all solutions σ, if $s_1 \cup \cdots \cup s_m \in \sigma(s)$ and $t_1 \cup \cdots \cup t_n \in \sigma(t)$, where all the s_i, t_j are atoms, then there are an atom a and an index i such that $s_i = a$ and for all t_j $t_j \neq a$ or, vice versa, $t_i = a$ and for all s_j $s_j \neq a$.*

Proof. Assume $\sigma(s) \neq \sigma(t)$. This means for all s' term in $\sigma(s)$ and t' term in $\sigma(t)$ it holds that $ACI1 \not\models (s' = t')$. But if all the atoms composing s' were in t' and vice versa, then by ACI we would have the equality. □

The structure $\mathcal{H}$ is also a model of the freeness equational axioms:

$$\begin{array}{lll}
(F_1) & f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n) \to X_1 = Y_1 \wedge \cdots \wedge X_n = Y_n & f \in \Sigma, f \not\equiv \cup \\
(F_2) & f(X_1, \ldots, X_m) \neq g(Y_1, \ldots, Y_n) & f \not\equiv g, f, g \not\equiv \cup \\
(F_3) & (X \neq t_1 \cup \cdots \cup f(\cdots X \cdots) \cup \cdots \cup t_n) & f \not\equiv \cup
\end{array}$$

The standard freeness axioms [9] have been here refined in order to capture the behavior of $\cup$.

Similarly to what we did for (D_c) in the previous section, we need to enforce the property that unions of distinct atoms return distinct objects of the domain. Instead of extending (D_c)—which would be quite cumbersome in this context—we introduce the following axiom scheme: for all $f \in \Sigma$, $f \not\equiv \cup$, $ar(f) = n$:

$$\begin{array}{ll}
(D_f) & f(X_1, \ldots, X_n) \cup X = Y_1 \cup Y_2 \to \\
& \exists Z_1, Z_2 \left(\begin{array}{l}
(Y_1 = f(X_1, \ldots, X_n) \cup Z_1 \wedge X = Z_1 \cup Y_2) \quad \vee \\
(Y_2 = f(X_1, \ldots, X_n) \cup Z_2 \wedge X = Y_1 \cup Z_2) \quad \vee \\
(Y_1 = f(X_1, \ldots, X_n) \cup Z_1 \wedge Y_2 = f(X_1, \ldots, X_n) \cup Z_2 \wedge X = Z_1 \cup Z_2)
\end{array} \right)
\end{array}$$

Observe that the $\leftarrow$ direction is a simple consequence of axioms ACI. This axiom scheme captures the intuitive notion of atoms in the context of $\cup$, and subsumes the case covered by (D_c). Thus, axiom (D_c) can be safely removed:

Lemma 4.9 $ACI1 D_f F_2 \models (D_c)$.[5]

Proof. We first prove three properties. For all $n > 0$:

$$(a) \qquad ACI1 \models \forall X_1 \cdots X_n (X_1 \cup \cdots \cup X_n = \emptyset \leftrightarrow X_1 = \emptyset \wedge \cdots \wedge X_n = \emptyset)$$

As a matter of fact, $X_1 \cup \cdots \cup X_n = \emptyset$ implies that $X_1 \cup (X_1 \cup \cdots \cup X_n) = X_1 \cup \emptyset = X_1$, and $(X_1 \cup X_1) \cup X_2 \cup \cdots \cup X_n = X_1 \cup X_2 \cup \cdots \cup X_n = \emptyset$. Similarly for $X_2, \ldots, X_n$. The converse is immediate.

$$(b) \qquad ACI1 D_f \models \forall X_1 \cdots X_n \left(X_1 \cup \cdots \cup X_n = c \leftrightarrow \bigwedge_{i=1}^{n} (X_i = c \vee X_i = \emptyset) \wedge \bigvee_{i=1}^{n} X_i = c \right)$$

By induction on $n > 0$. *Base:* $n = 1$: trivial.
Step: By (D_f),

$$c \cup \underbrace{\emptyset}_{X} = \underbrace{X_1}_{Y_1} \cup \underbrace{X_2 \cup \cdots \cup X_n}_{Y_2}$$

holds if and only if

1. $X_1 = c \cup Z_1 \wedge \emptyset = Z_1 \cup X_2 \cup \cdots \cup X_n$. By property (a), $Z_1 = X_2 = \cdots = X_n = \emptyset$. Then, the base case on the first equation forces $X_1 = c$.

[5] The converse is not true. Consider $\Sigma = \{\emptyset, \cup, c\}$ and the lattice $\perp < a_1, \perp < a_2, a_1 < c, a_2 < c^{\mathcal{A}}$. It fulfills (D_c) but not (D_f) (consider $c \cup \underbrace{\perp}_{X} = \underbrace{a_1}_{Y_1} \cup \underbrace{a_2}_{Y_2}$).

2. $X_2 \cup \cdots \cup X_n = c \cup Z_2 \wedge \emptyset = X_1 \cup Z_2$. By property (a), $Z_1 = X_1 = \emptyset$. By inductive hypothesis on $X_2 \cup \cdots \cup X_n = c$, it follows that $X_i = \emptyset$ or $X_i = c$, for $i = 2, \ldots n$ but at least one of them is c.

3. $X_1 = c \cup Z_1 \wedge X_2 \cup \cdots \cup X_n = c \cup Z_2 \wedge \emptyset = Z_1 \cup Z_2$. By property (a), $Z_1 = Z_2 = \emptyset$. By base case $X_1 = c$ and by inductive hypothesis $X_i = \emptyset$ or $X_i = c$, for $i = 2, \ldots n$.

If $a_1, \ldots, a_m, b_1, \ldots, b_n$ are constant symbols from Σ,

$$(c) \qquad ACI1D_f \models \left(a_1 \cup \cdots \cup a_m = b_1 \cup \cdots \cup b_n \leftrightarrow \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} a_i = b_j \wedge \bigwedge_{j=1}^{n} \bigvee_{i=1}^{m} a_i = b_j \right)$$

in other words, the "sets" $\{a_1, \ldots, a_m\}$ and $\{b_1, \ldots, b_n\}$ are equal. We will make use of set theoretic equality to denote formulae of the kind above. If $m = 0$ then $a_1 \cup \ldots \cup a_m = \emptyset$, similarly for n.

We prove the property by induction on $h = \min\{m, n\}$.

Base: $h = 0$. It is property (a).

Step: Assume $m, n > 0$. By (D_f) it holds that

$$a_1 \cup \underbrace{a_2 \cup \cdots \cup a_m}_{X} = \underbrace{b_1}_{Y_1} \cup \underbrace{b_2 \cup \cdots \cup b_n}_{Y_2}$$

is equivalent to

1. $b_1 = a_1 \cup Z_1 \wedge a_2 \cup \cdots \cup a_m = Z_1 \cup b_2 \cup \cdots \cup b_n$. By properties (a) (that excludes $a_1 = \emptyset$) and (b), $a_1 = b_1$ and ($Z_1 = \emptyset$ or $Z_1 = b_1$). By inductive hypothesis on the second equation, this choice is equivalent to the formula:

   (a) $a_1 = b_1 \wedge \{a_2, \ldots, a_m\} = \{b_1, \ldots, b_n\}$, or

   (b) $a_1 = b_1 \wedge \{a_2, \ldots, a_m\} = \{b_2, \ldots, b_n\}$,

2. $b_2 \cup \cdots \cup b_n = a_1 \cup Z_2 \wedge a_2 \cup \cdots \cup a_m = b_1 \cup Z_2$. By inductive hypothesis on both the equations (the condition is on the minimum) it holds that:

   (a) $Z_2 = \{b_2, \ldots, b_n\}$ or $Z_2 = \{b_2, \ldots, b_n\} \setminus \{a_1\}$, and

   (b) $Z_2 = \{a_2, \ldots, a_m\}$ or $Z_2 = \{a_2, \ldots, a_m\} \setminus \{b_1\}$.

   Thus, by applying distributivity, we get 4 sub-cases.

3. $b_1 = a_1 \cup Z_1 \wedge b_2 \cup \cdots \cup b_n = a_1 \cup Z_2 \wedge a_2 \cup \cdots \cup a_m = Z_1 \cup Z_2$. By properties (a) and (b), $a_1 = b_1$ and ($Z_1 = \emptyset$ or $Z_1 = a_1$). By inductive hypothesis on the second equation, this choice is equivalent to the formula $a_1 = b_1$ and

   (a) $Z_2 = \{b_2, \ldots, b_n\} \setminus \{a_1\}$ or $Z_2 = \{b_2, \ldots, b_n\}$, and

   (b) $Z_2 = \{a_2, \ldots, a_m\}$ or $Z_2 = \{a_2, \ldots, a_m\} \setminus \{b_1\}$.

   Also in this case, by applying distributivity, we get 4 sub-cases.

It is a matter of routine to verify the set-theoretic equivalence between $\{a_1, \ldots, a_m\} = \{b_1, \ldots, b_n\}$ and the disjunction of the ten formulae above when $m, n > 0$.

The proof of the proposition can be obtained from property (c) using axiom (F_2) that, together with equality, states that $a_i = b_j$ if and only if they are syntactically equal. □

We will introduce definitions and Lemmas to prove the correspondence between $\mathcal{H}$ and $ACI1F_1F_2F_3D_f$ on the class of all the conjunction of equations and disequations.

Given a first order language $\mathcal{L}$ and two structures $\mathcal{A}$ and $\mathcal{B}$ over $\mathcal{L}$ an embedding of $\mathcal{A}$ in $\mathcal{B}$ is an isomorphism from $\mathcal{A}$ into a substructure of $\mathcal{B}$. If the only predicative symbol of $\mathcal{L}$ is the equality, an embedding of $\mathcal{A}$ in $\mathcal{B}$ can be equivalently defined as an injective homomorphism from $\mathcal{A}$ to $\mathcal{B}$, [25].

Lemma 4.10 *Let $\mathcal{A}$ and $\mathcal{B}$ be two structures of a first order language $\mathcal{L}$, and let h be an embedding of $\mathcal{A}$ in $\mathcal{B}$. If φ is an unquantified (open) formula of $\mathcal{L}$, then*

$$\mathcal{A} \models \varphi[\sigma] \leftrightarrow \mathcal{B} \models \varphi[h \circ \sigma].$$

Proof. It can easily be proved by induction on the complexity of φ. In [25] the result is presented as a corollary of the first theorem of chapter 2. □

Lemma 4.11 *$\mathcal{H}$ is a model of the theory $ACI1F_1F_2F_3D_f$.*

Proof. $\mathcal{H}$ is a model of $ACI1$, since for any equational theories E, $T(\Sigma)/=_E$ is a model of E, [26]. If $f(t_1,\dots,t_n)$ and $f(s_1,\dots,s_n)$ are ground terms, then the only way to deduce $f(t_1,\dots,t_n) = f(s_1,\dots,s_n)$ from $ACI1$ is to deduce $\wedge_{i=1}^n(t_i = s_i)$ from $ACI1$, and then apply the logical axiom $\forall X, Y(X = Y \to t = t[X/Y])$ as last step. From this we obtain that $\mathcal{H}$ is also a model of the freeness axioms (by induction on the complexity of the terms).

We prove that $\mathcal{H}$ is a model of (D_f). If

$$\mathcal{H} \models f(X_1,\dots,X_n) \cup X = Y_1 \cup Y_2[\sigma],$$

then, from Lemma 4.8, the atom $f(X_1,\dots,X_n)\sigma$ must be an atom in $Y_1\sigma$ or in $Y_2\sigma$. This means that

$$\mathcal{H} \models (Y_1 = f(X_1,\dots,X_n) \cup W_1 \vee Y_2 = f(X_1,\dots,X_n) \cup W_2)[\sigma].$$

Since $\mathcal{H}$ is a model of $ACI1$, we can suppose that $f(X_1,\dots,X_n)\sigma$ is neither an atom of $W_1\sigma$ nor of $W_2\sigma$.

Let us suppose, as first case, that

$$\mathcal{H} \models \exists W_1(Y_1 = f(X_1,\dots,X_n) \cup W_1)[\sigma], \text{ and } \mathcal{H} \not\models \exists W_2(Y_2 = f(X_1,\dots,X_n) \cup W_2)[\sigma].$$

Since $\mathcal{H}$ is a model of (F_1), all the atoms of $(Y_1 \cup Y_2)\sigma$ that are different from $f(X_1,\dots,X_n)\sigma$, by Lemma 4.8, are atoms of $X\sigma$. If $f(X_1,\dots,X_n)\sigma$ is not an atom of $X\sigma$, then, always from Lemma 4.8 all the atoms of $W_1\sigma$ and $Y_2\sigma$ have to be atoms of $X\sigma$, and all the atoms of $X\sigma$ have to be atoms of $W_1\sigma$ or of $Y_2\sigma$, hence, from $ACI1$, $\mathcal{H} \models (X = W_1 \cup Y_2)\sigma$, i.e.

$$\mathcal{H} \models (Y_1 = f(X_1,\dots,X_n) \cup W_1 \wedge X = W_1 \cup Y_2)[\sigma].$$

If $f(X_1,\dots,X_n)\sigma$ is an atom of $X\sigma$, then consider $Z_1 = W_1 \cup f(X_1,\dots,X_n)$. It holds $\mathcal{H} \models (Y_1 = f(X_1,\dots,X_n) \cup Z_1)[\sigma]$, and as in the previous case $\mathcal{H} \models (X = Z_1 \cup Y_2)\sigma$.

The second case, in which $\mathcal{H} \models \exists W_2(Y_2 = f(X_1,\dots,X_n) \cup W_2)[\sigma]$, but $\mathcal{H} \not\models \exists W_1(Y_1 = f(X_1,\dots,X_n) \cup W_1)[\sigma]$, is similar.

The third case, in which $\mathcal{H} \models \exists W_1(Y_2 = f(X_1,\dots,X_n) \cup W_1)[\sigma]$, $\mathcal{H} \models \exists W_2(Y_1 = f(X_1,\dots,X_n) \cup W_2)[\sigma]$, is similar with $Z_1 = W_1 \cup f(X_1,\dots,X_n)$, when $f(X_1,\dots,X_n)\sigma$ is an atom of $X\sigma$. □

Lemma 4.12 *If $\mathcal{B}$ is a model of $ACI1F_1F_2F_3D_f$, then the function*

$$\begin{array}{rccc} h: & \mathcal{H} & \longrightarrow & \mathcal{B} \\ & h([t]) & = & t^{\mathcal{B}} \end{array}$$

is an embedding of $\mathcal{H}$ in $\mathcal{B}$.

Proof. We will prove the following facts:

1. the definition of $h([t])$ does not depends on the choice of the representative of the class;
2. h is an homomorphism;
3. h is injective.

Since $\mathcal{L}$ does not contain predicative symbols, differents from $=$, (2) and (3) ensures that h is an embedding.

1. If $[t_1] = [t_2]$, then $ACI1 \models t_1 = t_2$. Hence in every model $\mathcal{M}$ of $ACI1$ it holds $\mathcal{M} \models t_1 = t_2$, so this must hold in $\mathcal{B}$, i.e. $t_1^{\mathcal{B}} = t_2^{\mathcal{B}}$.

2. $\forall f \in \Sigma$ it holds $h(f^{\mathcal{H}}([t_1], \ldots, [t_n]) = h([f(t_1, \ldots, t_n)]) = (f(t_1, \ldots, t_n))^{\mathcal{B}}$, since we have proved that h does not depend on the representatives of the classes. Moreover $(f(t_1, \ldots, t_n))^{\mathcal{B}} = f^{\mathcal{B}}(t_1^{\mathcal{B}}, \ldots, t_n^{\mathcal{B}})$, by the definition of structure, and $f^{\mathcal{B}}(t_1^{\mathcal{B}}, \ldots, t_n^{\mathcal{B}}) = f^{\mathcal{B}}(h([t_1]), \ldots, h([t_n]))$.

3. If $h([t_1]) = h([t_2])$, then we prove $[t_1] = [t_2]$. We have already proved that h does not depend on the representative of the class, hence we can assume that t_1 and t_2 are in canonical form. Using this assumption and Corollary 2.7 we have to prove that if t_1 and t_2 are in canonical form and $h([t_1]) = h([t_2])$, then $t_1 \equiv t_2$.

   We prove this fact by structural induction on t_1.

   Base: Let $t_1 \equiv c$ be constant.

   Since $\mathcal{B}$ is a model of F_2 it can not be $t_2 \equiv f(s_1, \ldots, s_n)$, with $f \not\equiv \cup, c$.

   Moreover $\mathcal{B}$ is a model of $ACI1D_f$, hence

   $$c^{\mathcal{B}} \cup \emptyset^{\mathcal{B}} = s_1^{\mathcal{B}} \cup s_2^{\mathcal{B}} \text{ implies } (s_1^{\mathcal{B}} = c^{\mathcal{B}} \wedge s_2^{\mathcal{B}} = \emptyset^{\mathcal{B}}) \vee (s_1^{\mathcal{B}} = \emptyset^{\mathcal{B}} \wedge s_2^{\mathcal{B}} = c^{\mathcal{B}}) \vee (s_1^{\mathcal{B}} = c^{\mathcal{B}} \wedge s_2^{\mathcal{B}} = c^{\mathcal{B}})$$

   and since $\mathcal{B}$ is a model of F_2 this implies

   $$(s_1 \equiv c \wedge s_2 \equiv \emptyset) \vee (s_1 \equiv \emptyset \wedge s_2 \equiv c) \vee (s_1 \equiv c \wedge s_2 \equiv c)$$

   Thus, since t_2 is in canonical form, it can not be the case that $t_2 \equiv s_1 \cup s_2$ (this can be generalized to obtain $t_2 \not\equiv s_1 \cup s_2 \cup s_n$).

   Step:

   Let $t_1 \equiv f(s_1, \ldots, s_n)$, with $f \not\equiv \cup$.

   It can not be $t_2 \equiv g(r_1, \ldots, r_m)$, with $g \not\equiv f, \cup$, since $\mathcal{B}$ is a model of F_2.

   As in the previous case we obtain that t_2 can not be of the form $r_1 \cup \ldots \cup r_m$.

   So, it must be $t_2 \equiv f(r_1, \ldots, r_n)$, and $s_i^{\mathcal{B}} = r_i^{\mathcal{B}}$, for all $i \leq n$. Using the inductive hypothesis we have $t_1 \equiv t_2$.

   Let $t_1 \equiv s_1 \cup \ldots \cup s_m$.

   It can not be $t_2 \equiv f(r_1, \ldots, r_n)$ (from the previous case applied to t_2).

   It must be $t_2 \equiv r_1 \cup \ldots \cup r_n$.

   Since t_1 and t_2 are in canonical form, it must be $s_i \equiv f_i(\ldots)$ and $r_j \equiv g_j(\ldots)$, with $f, g \not\equiv \cup$. By contraddiction if $t_1^{\mathcal{B}} \not\equiv t_2^{\mathcal{B}}$, since $\mathcal{B}$ is a model of $ACI1$ it must be:

   $$\exists i \leq m \, \forall j \leq n \; s_i^{\mathcal{B}} \not\equiv t_j^{\mathcal{B}} \vee \exists j \leq n \; i \leq m \; t_j^{\mathcal{B}} \not\equiv s_i^{\mathcal{B}}$$

   without loss of generality we can consider to be in the first case of the disjunction, and $i = 1$. By inductive hypothesis we have $\forall j \leq n \; h([s_1]) \neq h([t_j])$. Hence, since we have proved h is an homomorphism it holds

   $$\begin{array}{rcl} h([t_1]) & = & h([s_1]) \cup^{\mathcal{B}} \ldots \cup^{\mathcal{B}} h([s_m]) \\ & = & s_1^{\mathcal{B}} \cup^{\mathcal{B}} \ldots \cup^{\mathcal{B}} s_m^{\mathcal{B}} \\ h([t_2]) & = & h([r_1]) \cup^{\mathcal{B}} \ldots \cup^{\mathcal{B}} h([r_n]) \\ & = & r_1^{\mathcal{B}} \cup^{\mathcal{B}} \ldots \cup^{\mathcal{B}} r_n^{\mathcal{B}} \end{array}$$

   From this, since $\mathcal{B}$ is a model of D_f and $s_i \equiv f_1(\ldots)$, we have $h([t_1]) \neq h([t_2])$.

□

Theorem 4.13 *$\mathcal{H}$ and $ACI1F_1F_2F_3D_f$ correspond on the class of all the conjunctions of equations and disequations.*

Proof. From Lemma 4.11 it follows that if C is a first order formulae and $ACI1F_1F_2F_3D_f \models C$, then $\mathcal{H} \models C$.

On the other hand if $\vec{\exists} C$ is a formula with only existential quantifiers, then $\mathcal{H} \models \vec{\exists} C$ if and only if there exists σ such that $\mathcal{H} \models C[\sigma]$, hence from the Lemma 4.12 (and from the Lemma 4.10) it holds $\mathcal{B} \models \vec{\exists} C$ for all $\mathcal{B}$ models of $ACI1F_1F_2F_3D_f$, but this means $ACI1F_1F_2F_3D_f \models \vec{\exists} C$. □

4.3.2 Satisfiability

The theorem we present in this section is fundamental for the satisfiability test of all the solved forms presented in this paper.

A valuation $\sigma : \mathcal{V} \mapsto \mathcal{H}$ is called *atomic* if for all $X \in \mathcal{V}$, $X\sigma$ is an atom of $\mathcal{H}$. Note that from Lemma 4.7 we have that if σ is atomic, then $X\sigma$ is a congruence class of the form $[f(t_1, \dots, t_n)]$, with $f \not\equiv \emptyset, \cup$.

Lemma 4.14 *Let Σ be a general signature, and let s and $t \equiv t_1 \cup \ldots \cup t_n$ be two normalized terms, s and t_i all atoms.*

1. *If $s \not\equiv t$, then there exists $i \leq n$ such that $s \not\equiv t_i$.*
2. *If σ is an atomic valuation which satisfies $\bigvee_{i=1}^{n}(s \neq t_i)$, then σ satisfies $s \neq t$.*

Proof. 1) If for all $i \leq n$ we have $s \equiv t_i$, then it is also true that for all $i, j \leq n$ $t_i \equiv t_j$ and hence, since t is normalized, it must be $t \equiv t_1$. From $t \equiv t_1$ and $s \equiv t_1$ we obtain $s \equiv t$, which contradicts the initial assumption $s \not\equiv t$.

2) Let us show this results for $n = 2$—the proof can be immediately generalized to arbitrary n. Without loss of generality we can assume that σ satisfies $s \neq t_1$— from ACI. From (D_f) we have

$$s \cup \emptyset = t_1 \cup Y \leftrightarrow \exists A, B \left(\begin{array}{l} (t_1 = s \cup A \wedge \emptyset = A \cup Y) \vee \\ (Y = s \cup B \wedge \emptyset = B \cup t_1) \vee \\ (t_1 = s \cup A \wedge Y = s \cup B \wedge \emptyset = A \cup B) \end{array} \right)$$

By property (a) in proof of Lemma 4.9 we have $\emptyset = A \cup B \leftrightarrow A = \emptyset \wedge B = \emptyset$. This allows us to rewrite the above formula as

$$s = t_1 \cup Y \leftrightarrow (t_1 = s \wedge Y = \emptyset) \vee (Y = s \wedge t_1 = \emptyset) \vee (t_1 = s \wedge Y = s)$$

Since t is a normalized term then also t_1 is a normalized term and $t_1 \not\equiv \emptyset$, so from (F_2) we can infer that $t_1 = \emptyset$ is unsatisfiable. Hence, we are left with

$$s = t_1 \cup Y \leftrightarrow (t_1 = s \wedge Y = \emptyset) \vee (t_1 = s \wedge Y = s)$$

In our case $s\sigma \neq t_1\sigma$ which leads to $s\sigma \neq t_1\sigma \cup Y$ for any Y, and hence $s\sigma \neq t_1\sigma \cup t_2\sigma$. □

Lemma 4.15 *Let Σ be a general signature. Let $s \equiv s_1 \cup \ldots \cup s_m$ and $t \equiv t_1 \cup \ldots \cup t_n$ be two normalized terms, s_i, t_j all atoms.*

1. *If $s \not\equiv t$, then:*
   - *there is $j \leq m$ such that, for all $i \leq n$, the relation $s_j \not\equiv t_i$ holds, or*
   - *there is $i \leq n$ such that, for all $j \leq m$, the relation $t_i \not\equiv s_j$ holds.*

2. *Assume that σ is an atomic valuation which satisfies* $(\bigvee_{j=1}^{m}(\bigwedge_{i=1}^{n}(s_j \neq t_i))) \vee (\bigvee_{i=1}^{n}(\bigwedge_{j=1}^{m}(t_i \neq s_j)))$, *then σ satisfies $s \neq t$.*

Proof. 1) Let us assume that for each $j \leq m$ there exists $i_j \leq n$ such that $s_j \equiv t_{i_j}$. Let us also assume that for each $i \leq n$ there exists $j_i \leq m$ such that $t_i \equiv s_{j_i}$. Since s and t are in normalized form it, these assumptions leads to $m = n$ and for all j and i $i = j_i$ and $j = i_j$. As a consequence $s \equiv t$ must hold, which contradicts the lemma's hypothesis.

2) Let us show this result for $m = 2, n = 2$—the proof can be immediately generalized to arbitrary m, n. Without loss of generality we can assume that σ satisfies $\wedge_{i=1}^{n}(s_1 \neq t_i)$— from ACI. In our case $s_1 \neq t_1 \wedge s_1 \neq t_2$.

From (D_f) we have

$$s_1 \cup Y = t_1 \cup t_2 \leftrightarrow \exists A, B \left(\begin{array}{l} (t_1 = s_1 \cup A \wedge Y = A \cup t_2) \vee \\ (t_2 = s_1 \cup B \wedge Y = B \cup t_1) \vee \\ (t_1 = s_1 \cup A \wedge t_2 = s_1 \cup B \wedge Y = A \cup B) \end{array} \right)$$

Notice that $s_1 \cup s_2$ and $t_1 \cup t_2$ are normalized and hence s_1, s_2, t_1, t_2 are normalized and different from $\emptyset$. This allows us to reuse the same reasoning of the proof of Lemma 4.14.2. Since $s_1\sigma \neq t_1\sigma$ from the proof of lemma 4.14.2 we have that $t_1\sigma \neq s_1\sigma \cup A$ holds for all A. In the same way, since $s_1\sigma \neq t_2\sigma$ we have that $t_2\sigma \neq s_1\sigma \cup B$ holds for all B. So it must be that $s_1\sigma \cup Y \neq t_1\sigma \cup t_2\sigma$ for all Y, and hence, with $Y = s_2\sigma$, we obtain $s\sigma \neq t\sigma$. □

We define the function *depth* : $T(\Sigma) \longrightarrow \mathbb{N}$ as:

$$\begin{cases} depth(\emptyset) &= 0 \\ depth(c) &= 1 & \text{if } c \text{ is a constant, } c \not\equiv \emptyset \\ depth(t_1 \cup t_2) &= \max_{i=1}^{2} depth(t_i) \\ depth(f(t_1, \ldots, t_n)) &= 1 + \max_{i=1}^{n} depth(t_i) & f \not\equiv \cup \end{cases}$$

Given a general signature Σ, we define the structure $\mathcal{N} = \langle \mathbb{N}, (\cdot)^{\mathcal{N}} \rangle$, where

$$\begin{cases} \emptyset^{\mathcal{N}} &= 0 \\ c^{\mathcal{N}} &= 1 & \forall c \in \Sigma, c \text{ constant, } c \not\equiv \emptyset \\ n_1 \cup^{\mathcal{N}} n_2 &= \max\{n_1, n_2\} \\ f(n_1, \ldots, n_m)^{\mathcal{N}} &= 1 + \max\{n_1, \ldots, n_m\} & \forall f \in \Sigma, f \not\equiv \cup \end{cases}$$

Lemma 4.16 *The function depth is an homomorphism from $\mathcal{H}$ to $\mathcal{N}$. Moreover, if $\mathcal{N} \models (s \neq t) depth \circ \sigma$, then $\mathcal{H} \models (s \neq t)\sigma$.*

Proof. We prove that *depth* is an homomorphism:

1. $depth(\emptyset^{\mathcal{H}}) = 0 = \emptyset^{\mathcal{N}}$;
2. $depth(c^{\mathcal{H}}) = 1 = c^{\mathcal{N}}$;
3. $depth(t_1 \cup^{\mathcal{H}} t_2) = \max\{depth(t_1), depth(t_2)\} = depth(t_1) \cup^{\mathcal{H}} depth(t_2)$;
4. $depth(f^{\mathcal{H}}(t_1, \ldots, t_n)) = 1 + \max_{i=1}^{n} depth(t_i) = f^{\mathcal{N}}(depth(t_1), \ldots, depth(t_n))$.

By [25], if σ satisfies $s = t$ in $\mathcal{H}$, since *depth* is an homomorphism, then $depth \circ \sigma$ satisfies $s = t$ in $\mathcal{N}$. This allows to derive the second part of the proposition by contradiction. □

Theorem 4.17 *Let Σ be a general signature. Given a disequation constraint $C \equiv (s_1 \neq t_1 \wedge \cdots \wedge s_n \neq t_n)$ such that $\rho(s_i) \not\equiv \rho(t_i)$ for all $i = 1, \ldots, n$, then C is satisfiable in $\mathcal{H}$, and in every model of $ACI1F_1F_2F_3D_f$.*

Proof. First of all, according to Corollary 2.8, we have that $ACI1 \models \vec{\forall}(C \leftrightarrow \rho(C))$. So without loss of generality we can assume every s_i, t_i in C are normalized terms.

To achieve the desired result, we proceed as follows: we start by building a new system R, we show that each atomic valuation σ satisfying R should satisfy C as well, and finally we prove that R admits always at least one atomic solution.

If $(s_i \neq t_i) \in C$ and both s_i and t_i are not of the form $\_ \cup \_$, then let us consider the disequation $s_i \neq t_i$ in R. Obviously each solution of R is also a solution of $s_i \neq t_i$.

If $(s_i \neq t_{i_1} \cup \ldots \cup t_{i_k}) \in C$, with s_i and the t_{i_j}'s not of the form $\_ \cup \_$, then since $s_i \not\equiv t_{i_1} \cup \ldots \cup t_{i_k}$ (by hypothesis), there must exists $j \leq h$ such that $s_i \not\equiv t_{i_j}$—from lemma 4.14.1. Hence let us consider the disequation $s_i \neq t_{i_j}$ in R. A atomic solution σ for R is always a atomic solution for $s_i \neq t_i$, since $s_i\sigma$ is an atom different from $t_{i_j}\sigma$ (from Lemma 4.14.2).

If $(s_{i_1} \cup \ldots \cup s_{i_h} \neq t_{i_1} \cup \ldots \cup t_{i_k}) \in C$, with the s_{i_ℓ} and the t_{i_j} not of the form $\_ \cup \_$, then from Lemma 4.15.1 and without loss of generality we can assume that $\wedge_{j=1}^{k}(s_{i_1} \not\equiv t_{i_j})$. Hence, let us consider the disequations $s_{i_1} \neq t_{i_1}, \ldots, s_{i_1} \neq t_{i_k}$ in R. An atomic solution σ for R is a solution for $s_i \neq t_i$, from Lemma 4.15.2.

So, if we prove that R always admits a atomic solution σ, then we will be able to conclude the desired result. For doing that we introduce the auxiliary function $\mathit{find} : T(\Sigma, \mathcal{V}) \times \mathcal{V} \longrightarrow \wp(\mathbb{N})$.

$$\mathit{find}(t, X) = \begin{cases} \emptyset & \text{if } X \text{ does not occur in } t \\ \{0\} & \text{if } t \equiv X \\ \{n \mid n \in \mathit{find}(t_1, X) \vee n \in \mathit{find}(t_2, X)\} & \text{if } t \equiv t_1 \cup t_2 \\ \{n+1 \mid n \in \mathit{find}(t_i, X) \wedge 1 \leq i \leq m\} & \text{if } t \equiv f(t_1, \ldots, t_m) \end{cases}$$

For every variable X in R consider the auxiliary variable N_X, which ranges over integer numbers. Additionally, for each

- pair of terms $\langle s, t \rangle$ occurring in R such that $s \not\equiv t$
- variable X in s
- variable Y in t such that $Y \not\equiv X$
- $m_{(s,X)}$ in $\mathit{find}(s, X)$
- $m_{(t,Y)}$ in $\mathit{find}(t, Y)$

we introduce the disequation

$$N_X \neq N_Y + m_{(t,Y)} - m_{(s,X)}.$$

For every variable N_X consider also the disequation

$$N_X > \max\{\mathit{depth}(v) \mid v \in R, v \text{ ground}\},$$

where $v \in R$ stands for v is a term which occurs in R. This system of disequations admits always a solution. Let us indicate with $\{n_X \mid X \in \mathit{vars}(R)\}$ a solution of such system.

Let r^h be the term constructed as follows:

$$\begin{cases} r^0 & = & \emptyset \\ r^{k+1} & = & f(r^k, \ldots, r^k) \end{cases}$$

and consider the substitution $\sigma = \{X/r^{n_X} \mid X \in \mathit{vars}(R)\}$. We can prove that σ satisfies R. It is easy to see that $\mathit{depth}(X\sigma) = n_X$.

We prove that if s, t are two terms that occur in R (subterms of the s_i, t_i), and $s \not\equiv t$, then $s\sigma \neq t\sigma$ in $\mathcal{H}$. From this result we can conclude the desired thesis, since if $s_i \neq t_i$ is in R then s_i, t_i are two terms which occur in R and $s_i \not\equiv t_i$.

Let us proceed by induction on the structure of s. Since $=$ is symmetric we can avoid to consider the case s, t if the case t, s has already been considered.

If $s \equiv \emptyset$, then, since $t \not\equiv s$, $\mathit{depth}(t\sigma) > 0 = \mathit{depth}(s\sigma)$, from Lemma 4.16, we have $s\sigma \neq t\sigma$. If, instead $s \equiv c$ with c constant, then, since $t \not\equiv s$, it can be:

- t is a constant different from c, and hence $s\sigma = s \neq t = t\sigma$;
- $t \equiv c_1 \cup \ldots \cup c_n$, all the c_i constants, then from $t \not\equiv s$ and Lemma 4.14.1, there exists $i \leq n$, such that $s \not\equiv c_i$, hence we have the thesis;
- t is a ground term different from a constant, a union of constants, or a non-ground term, in all these cases, since $n_X > 1$ for all X, $depth(t\sigma) > 1 = depth(s)$, so from Lemma 4.16 we have the thesis.

If $s \equiv X$, then by induction on t:

- if t is ground, then, since $depth(X\sigma) = n_X > depth(t)$, from lemma 4.16 we have the thesis;
- if $t \equiv Y$, then $n_X \neq n_Y$ is an equation of the system, hence from Lemma 4.16 we have the thesis;
- if $t \equiv f(t_1, \ldots, t_n)$ and X occurs in t, (F_3) leads to the thesis;
- if $t \equiv f(t_1, \ldots, t_n)$ and X does not occur in t, then since we have imposed $N_Y > max\{depth(v) \mid v \in R, v \text{ ground}\}$ for all Y, there must exists a variable Y in t such that $depth(t\sigma) = n_Y + m_{(t,Y)}$ with $m_{(t,Y)} \in find(t, Y)$, in the system we have imposed $n_X \neq n_Y + m_{(t,Y)}$, hence from Lemma 4.16 we have the thesis;
- if $t \equiv t_1 \cup \ldots \cup t_n$, then from Lemma 4.14.1, there exists t_i such that $t_i \not\equiv X$, and hence by induction $t_i\sigma \neq X\sigma$, so from Lemma 4.14.2 we have the thesis.

If $s \equiv f(s_1, \ldots, s_n)$, then by induction on t:

- the result is trivial if $t \equiv g(t_1, \ldots, t_m)$;
- if $t \equiv f(t_1, \ldots, t_n)$, then it must be $s_i \not\equiv t_i$ for some $i \leq n$. By inductive hypothesis we obtain $s_i\sigma \neq t_i\sigma$, and from (F_1) we have the result;
- if $t \equiv t_1 \cup \ldots \cup t_m$, then, from Lemma 4.14.1 there is $t_i \not\equiv s$. By inductive hypothesis on t_i, we have that $s\sigma \neq t_i\sigma$, and from Lemma 4.14.2 we can obtain the result.

If $s \equiv s_1 \cup \ldots \cup s_n$, and $t \equiv t_1 \cup \ldots \cup t_m$ (this is the only case not yet considered), then from Lemma 4.15.1 we can focus on $\wedge_{i=1}^{m}(s_1 \not\equiv t_i)$. By inductive hypothesis $\wedge_{i=1}^{m}(s_1\sigma \neq t_i\sigma)$ holds, and from Lemma 4.15.2 we can obtain the desired result.

Since we have proved that C is satisfiable in $\mathcal{H}$, from Theorem 4.13, we have that C is satisfiable in every model of $ACI1F_1F_2F_3D_f$. □

Corollary 4.18 *If Σ is a general signature, then the satisfiability of a disequation constraint in $\mathcal{H}$ can be decided in polynomial time.*

5 Solved forms

Most constraint systems rely on the availability of *constraints simplifiers* to transform constraints into equivalent "simpler" formulae. In particular, it is common to identify a class of formulae, called *solved forms*, which are the target of this simplification. As described in [11], solved forms should be:

- **solvable:** each solved form is either false or it admits at least one solution;
- **simple:** satisfiability of a solved form should be *trivially* decidable;
- **complete:** every constraint is equivalent to a finite (possibly empty) disjunction of solved forms.

We propose three solved forms for the $ACI1$-constraints considered in this paper. Each form can be incrementally computed from the previous ones. The first form (*implicit*) represents implicitly its set of solutions. Given a constraint, a unique implicit solved form constraint can be computed from it in polynomial (quadratic) time. The second solved form (*intermediate*) further simplifies a constraint in implicit

function impl_simpl(φ):
while there is a disequation c in φ not in implicit s.f. do
case c of

(1)	$r \neq r$	$\mapsto$	false
(2)	$s \neq t$ $s \not\equiv t, vars(s) = \emptyset \wedge vars(t) = \emptyset$	$\mapsto$	true
(3)	$t \neq X$, t is not a variable	$\mapsto$	$X \neq t$
(4)	$X \neq t_1 \cup \cdots \cup f(\cdots X \cdots) \cup \cdots \cup t_n$, $f \not\equiv \cup$	$\mapsto$	true
(5)	$f(\cdots) \neq g(\cdots)$, $f \not\equiv g, f \not\equiv \cup, g \not\equiv \cup$	$\mapsto$	true

Figure 4: Rewriting procedure for implicit solved form

normal form. In polynomial (cubic) time it is possible to compute a formula containing a collection of intermediate normal form constraints. Each component of the collection can be computed in polynomial time, but the number of components might be exponential (exponentiality arises due to the application of distributivity). The third solved form (*explicit*) represents explicitly its solutions. It can be computed from the first one on demand. In this case, determining each disjunct requires exponential time.

All the solved forms are instances of the *compact formulae* defined in [11]. If the theory $ACI1$ were *compact*, the ability to reach a solved form would automatically ensure satisfiability of these constraints over $\mathcal{H}$ [11]. Unfortunately (see Sect. 6.3), $ACI1$ is not compact; thus, a direct proof of satisfiability of the different solved forms is required.

In the following subsections we precisely characterize the three solved forms. However, we anticipate that all of them fulfill the hypothesis of Theorem 4.17—thus, we have:

Corollary 5.1 *An implicit/intermediate/explicit solved form constraint C different from* false *is satisfiable in $\mathcal{H}$, and hence $ACI1F_1F_2F_3D_f \models \vec{\exists}C$.*

5.1 Implicit solved form

A variable X occurs *nested* in a term t if t is of the form: $t_1 \cup \cdots \cup f(\cdots X \cdots) \cup \cdots \cup t_n$, $f \not\equiv \cup$. A constraint C is in *implicit solved form* if it is false, true, or $C \equiv (s_1 \neq t_1 \wedge \cdots \wedge s_n \neq t_n)$ and for all $i = 1, \ldots, n$

- $vars(s_i) \cup vars(t_i) \neq \emptyset$, and
- $s_i \equiv \rho(s_i)$ and $t_i \equiv \rho(t_i)$, and
- $s_i \not\equiv t_i$, and
- if t_i is a variable, then s_i is a variable, as well, and
- if s_i is a variable, then it does not occur nested in t_i.

Given a constraint $C \equiv (s_1 \neq t_1 \wedge \cdots \wedge s_n \neq t_n)$, we can obtain an equivalent constraint in implicit solved form, starting from $\rho(C)$ (cf. Sect. 2.3) and applying the function impl_simpl of Fig. 4. Occurrences of true and false can be easily handled in linear time at the end of the process—we assume that the function tfsimpl performs this task. Clearly, tfsimpl(impl_simpl($\rho(t)$)) is in implicit normal form.

Proposition 5.2 *Given a constraint C, then* tfsimpl(impl_simpl($\rho(C)$)) *can be computed in time $O(n^2)$ where $n = |C|$. Moreover, if $\bar{X} = vars(C)$, then $ACI1F_1F_2F_3 \models \forall \bar{X}(C \leftrightarrow$ impl_simpl$(\rho(C)))$.*

Proof. Complexity follows from Prop. 2.6 and by the fact that the function impl_simpl analyze each disequation at most twice.

Correctness follows from Corollary 2.7 and by analyzing the relationships from each rewriting rule and axioms $F_1F_2F_3$. □

function int_simpl(φ): while there is a disequation c in φ not in intermediate s.f. do case c of			
(1)–(5) as in impl_simpl			
(6)	$\left.\begin{array}{r} s_1 \cup s_2, \neq f(t_1,\ldots,t_n), \\ f \not\equiv \cup \end{array}\right\}$	$\mapsto$	$f(t_1,\ldots,t_n) \neq s_1 \cup s_2$
(7)	$f(s_1,\ldots,s_n) \neq f(t_1,\ldots,t_n)$	$\mapsto$	$\bigvee_{i=1}^{n} s_i \neq t_i$
(8)	$f(s_1,\ldots,s_n) \neq t_1 \cup t_2$	$\mapsto$	$\bigvee_{i=1}^{2}(f(s_1,\ldots,s_n) \neq t_i \wedge t_i \neq \emptyset) \vee$ $\vee(f(s_1,\ldots,s_n) \neq t_1 \wedge f(s_1,\ldots,s_n) \neq t_2)$

Figure 5: Rewriting procedure for intermediate solved form

5.2 Intermediate solved form

A constraint in implicit solved form C is in *intermediate solved form* if it is false or $C \equiv (s_1 \neq t_1 \wedge \cdots \wedge s_n \neq t_n)$ and for all $i = 1,\ldots,n$

- $s_i \equiv X$ (and X does not occur nested in t_i), or
- $s_i \equiv r_1 \cup \cdots \cup r_h$ and $t_i \equiv v_1 \cup \cdots \cup v_k$, $h, k \geq 2$.

The procedure described in Fig. 5 produces a formula containing only disequations in intermediate solved form. Moreover, the formula is equivalent to the original constraint.

Proposition 5.3 *Given a constraint C, then the formula* int_simpl(C) *can be computed in time $O(n^3)$ where $n = |C|$. Moreover, $ACI1F_1F_2F_3D_f \models \vec{\forall}(C \leftrightarrow$ int_simpl$(C))$.*

Proof. The complexity bound follows from the analysis of a simplified version of the problem. It is easy to see that only a linear number of further actions can be produced by the complete case. Consider $\Sigma = \{\emptyset, f, \cup\}$, $ar(f) = 1$, and consider the following rules that generates labeled trees:

($R1$) for each not marked node μ labeled $f(s) \neq f(t)$, generate a unique "son" node labeled $s \neq t$. Mark μ.

($R2$) for each not marked node μ labeled $f(s) \neq t_1 \cup t_2$ or $t_1 \cup t_2 \neq f(s)$, generate two "son" nodes, labeled $f(s) \neq t_1$ and $f(s) \neq t_2$, respectively. Mark μ.

All the nodes labeled differently (e.g., $\emptyset \neq \emptyset, \emptyset \neq f(s), s \cup t \neq s' \cup t'$) are leaves.

We prove that the number of nodes of the tree with the root labeled $s \neq t$ is less than or equal to $|s||t|$, where $|r|$ denotes the number of occurrences of constant or function symbols in r, by induction on $n = |s||t|$.
Base: $n = 1$: $c \neq c$ is a leaf: trivial.
Step:

($R1$) By inductive hypothesis, the tree with root labeled $s \neq t$ has a number of nodes $n \leq |s||t|$. Then, the number of nodes of the tree with root labeled $f(s) \neq f(t)$ has $n + 1 \leq (|s| + 1)(|t| + 1)$ nodes.

($R2$) By inductive hypothesis, the trees with root labeled $s \neq t_i$ have a number of nodes $n_i \leq |s||t_i|$. Thus, the number of nodes of the tree with root labeled $s \neq t_1 \cup t_2$ or in $t_1 \cup t_2 \neq s$ has $n_1 + n_2 + 1$ nodes. Now, $n_1 + n_2 + 1 \leq |s||t_1| + |s||t_2| + 1 \leq |s|(|t_1| + |t_2| + 1)$, since $|s| \geq 1$.

Each node contains two terms that are subterms of the initial one. Thus the global complexity is $O(n^3)$.[6]

The second part can be proved by the analysis of the various rewriting rules that exactly apply axioms (F_1), (F_2), and (F_3). □

Nevertheless, a successive application of the distributivity can generate an exponential number of disjuncts. This consideration holds in any general equational theory E. For example, given a binary function symbol f and the constraint $f(X_1, Y_1) \neq f(V_1, Z_1) \wedge \cdots \wedge f(X_n, Y_n) \neq f(V_n, Z_n)$. After the application of the simplification rules in Fig. 5 we obtain (in time $O(n)$): $(X_1 \neq V_1 \vee Y_1 \neq Z_1) \wedge \cdots \wedge (X_n \neq V_n \vee Y_n \neq Z_n)$.

[6] Of course, by making use of structure-sharing techniques, the same algorithm can be more efficiently implemented.

```
function expl_simpl(φ);
while there is a disequation c in φ not in explicit s.f. do
    if c ≡ r ∪ s ≠ t ∪ u
        then replace c by ν(c)
        else replace c by int_simpl(c)
```

Figure 6: Rewriting procedure for explicit solved form

The corresponding disjunctive normal form contains 2^n disjuncts. On the other hand, the presence of an exponential number of disjuncts generated by the int_simpl procedure implies that a complete description of the whole set of solutions requires exponential amount of time:

Corollary 5.4 *Given a constraint C, then an intermediate solved form constraint C' that implies C can be computed in time $O(n^3)$, where $n = |C|$. Moreover, the disjunction of all the intermediate solved form constraints equivalent to C can be computed in $O(2^n)$.*

Proof. Each single disjunct can be computed in from int_simpl(C) in linear time on its size. Of course, the application of tfsimpl is needed to remove useless true or false □

Adopting a structure-sharing technique to implement int_simpl it is easy to lower the complexity to $O(n^2)$ (instead of $O(n^3)$).

5.3 Explicit solved form

In this section we show how a disequation $r \cup s \neq t \cup u$ occurring in a constraint in intermediate solved form can be further simplified.

An intermediate solved form constraint C is in *explicit solved form* if it is false or $C \equiv (s_1 \neq t_1 \wedge \cdots \wedge s_n \neq t_n)$ and if for all $i = 1, \ldots, n$ the following holds: if s_i is not a variable, then $s_i \neq t_i$ is of the form:

$$X \cup Y_1 \cup \cdots \cup Y_h \neq Y_1 \cup \cdots \cup Y_h \cup r_1 \cup \cdots \cup r_k$$

The process used to convert a constraint C in intermediate form (and different from false) into an equivalent constraint in explicit form is based on the recursive replacement of each conjunct of the form

$$c \equiv \underbrace{X_1 \cup \ldots \cup X_m \cup t_1 \cup \ldots \cup t_h}_{\ell} \neq \underbrace{Y_1 \cup \ldots \cup Y_n \cup s_1 \cup \ldots \cup s_k}_{r},$$

with the formula $\nu(c) \equiv \varphi_\ell \vee \varphi_r$, computed as: $\varphi_\ell \;\equiv\; \bigvee_{i=1}^{h} \varphi^i_{term} \vee \bigvee_{i=1}^{m} \varphi^i_{var}$ and

$$\begin{array}{rcl} \varphi^i_{term} & \equiv & \bigwedge_{j=1}^{k}(t_i \neq s_j) \;\wedge\; \bigwedge_{j=1}^{n}(Y_j \neq Y_j \cup t_i) \\ \varphi^i_{var} & \equiv & \bigwedge_{J \subseteq \{1,\ldots,k\}}(X_i \cup Y_1 \cup \ldots \cup Y_n \neq Y_1 \cup \ldots \cup Y_n \cup \bigcup_{r \in J} s_r) \end{array}$$

The definition of φ_r is perfectly symmetrical. Intuitively, φ^i_{term} asserts the fact that t_i does not belong to the set described by the r.h.s., while φ^i_{var} states that X_i is not a subset of the r.h.s.

The replacement of c with $\nu(c)$ can generate a number of new disequations (e.g., $s_1 \neq t_1$ with $s_1 \equiv f(\bar{t}_1)$ and $s_1 \equiv g(\bar{t}_2)$) that can be rewritten by the function int_simpl of Fig. 5. However, all the subproblems generated are of *smaller* size and therefore the function expl_simpl described in Fig. 6 will eventually terminate.

The next propositions formalize these results. Propositions 5.5 and 5.6 demonstrate that the transformation described in Fig. 6 produces a constraint which is equivalent to the original one—i.e., the solutions space is not modified. Proposition 5.7 shows that the algorithm in Fig. 6 terminates for any possible choice of initial constraints.

Proposition 5.5 *For a canonical disequation c of the form $X_1 \cup \ldots \cup X_m \cup t_1 \cup \ldots \cup t_h \neq Y_1 \cup \ldots \cup Y_n \cup s_1 \cup \ldots \cup s_k$ we have that $ACI1F_1F_2F_3D_f \;\models\; \vec{\forall}(c \leftrightarrow \nu(c))$.*

Proof. (Sketch) Let us analyze the simple case of $e \equiv (X \cup t \neq Y \cup s)$, s and t terms with outermost symbol different from $\cup$. The proof can be derived through generalization of this case. For this equation, the formula in the proposition reduces to:

$$\begin{array}{rll} \nu(e) \equiv & (s \neq t \wedge Y \cup t \neq Y) & \vee \\ & (X \cup Y \neq Y \wedge X \cup Y \neq Y \cup s) & \vee \\ & (s \neq t \wedge X \cup s \neq X) & \vee \\ & (X \cup Y \neq X \wedge X \cup Y \neq X \cup t) & \end{array}$$

If we complement this formula we obtain

$$\begin{array}{lcl} (s = t \vee Y \cup t = Y) & \wedge & (X \cup Y = Y \vee X \cup Y = Y \cup s) \wedge \\ (s = t \vee X \cup s = X) & \wedge & (X \cup Y = X \vee X \cup Y = X \cup t) \end{array} \tag{1}$$

We prove that $X \cup t = Y \cup s$ is indeed equivalent to formula (1).

It is easy to show that the formula (1) implies $X \cup t = Y \cup s$ by the analysis of each of the 16 disjuncts obtained by applying distributivity over (1). For instance, the first disjunct is: $s = t \wedge X \cup Y = Y \wedge s = t \wedge X \cup Y = X$, from which we have: $X \cup t = X \cup Y \cup t = X \cup Y \cup s = Y \cup s$.

To prove the converse, from (D_f) axiom we have that $X \cup t = Y \cup s$ if and only if:

$$\exists A, B \left(\begin{array}{ll} (Y = A \cup t \wedge X = A \cup s) & \vee \\ (s = B \cup t \wedge X = B \cup Y) & \vee \\ (Y = A \cup t \wedge s = t \cup B \wedge X = A \cup B) & \end{array} \right) \tag{2}$$

We can show that each disjunct of the formula (2) implies one disjunct in (1).

- $s = t \vee Y \cup t = Y$:
  1. $Y = A \cup t$ implies $Y \cup t = A \cup t \cup t = A \cup t$ which in turn implies $Y \cup t = Y$ (and, thus, $s = t \vee Y \cup t = Y$)
  2. $s = B \cup t$ implies $s = t$ (axiom (D_f)).
  3. $s = t \cup B$ implies $s = t$ (from axiom (D_f)).
- $X \cup Y = Y \vee X \cup Y = Y \cup s$:
  1. $Y = A \cup t \wedge X = A \cup s$ implies $X \cup Y = A \cup t \cup s = Y \cup s$, and $Y \cup s = A \cup t \cup s$. Thus, $X \cup Y = Y \vee X \cup Y = Y \cup s$
  2. $s = B \cup t \wedge X = B \cup Y$ implies $s = t \wedge (B = \emptyset \vee B = t) \wedge X = B \cup Y$ (axiom (D_f)); this implies $X = Y \vee X = Y \cup s$ which in turn implies $X \cup Y = Y \vee X \cup Y = Y \cup Y \cup s = Y \cup s$
  3. $Y = t \cup A \wedge s = t \cup B \wedge X = A \cup B$ implies

     $$s = t \wedge (B = \emptyset \vee B = t) \wedge X = A \cup B \wedge Y = A \wedge t$$

     this implies

     $$X \cup Y = A \cup t \cup B \wedge s = t \wedge (B = \emptyset \vee B = t) \wedge Y = A \wedge t$$

     which leads to

     $$X \cup Y = Y \vee X \cup Y = Y \cup s$$
- $s = t \vee X \cup s = X$:
  1. $Y = A \cup t \wedge X = A \cup s$ implies $X \cup s = A \cup s \cup s = A \cup s = X$ and thus $s = t \vee X \cup s = X$
  2. $s = B \cup t \wedge X = B \cup Y$ implies $s = t$ and thus $s = t \vee X \cup s = X$
  3. $Y = t \cup A \wedge s = t \cup B \wedge X = A \cup B$ implies $s = t$ and thus $s = t \vee X \cup s = X$

- $X \cup Y = X \vee X \cup Y = X \cup t$:

    1. $Y = A \cup t \wedge X = A \cup s$ implies $X \cup Y = A \cup s \cup t = X \cup t$ and thus $X \cup Y = X \vee X \cup Y = X \cup t$
    2. $s = B \cup t \wedge X = B \cup Y$ implies $s = t \wedge (B = \emptyset \vee B = t) \wedge X = B \cup Y$; this leads to $X \cup Y = Y \cup B \cup Y = Y \cup B = X$
    3. $Y = t \cup A \wedge s = t \cup B \wedge X = A \cup B$ implies $s = t \wedge (B = \emptyset \vee B = t) \wedge Y = A \cup t \wedge X = A \cup B$; this leads to

    $$X \cup Y = A \cup B \cup A \cup t = A \cup B \cup t = X \cup B \wedge (B = \emptyset \vee B = t)$$

    which finally leads to

    $$X \cup Y = X \vee X \cup Y = X \cup t$$

□

Proposition 5.6 *Given an implicit solved form constraint C, $ACI1F_1F_1F_2F_3D_f \models \vec{\forall}(C \leftrightarrow \mathsf{expl\_simpl}(C))$.*

Proof. This is an immediate consequence of Propositions 5.5 and 5.3. □

Theorem 5.7 *Given a implicit solved form constraint C, then $\mathsf{expl\_simpl}(C)$ terminates after the application of $O(n^3)$ steps, adopting a structure-sharing technique.*

Proof. Basically, the idea is the same as that used in proving Prop. 5.3. When a constraint c is replaced by $\nu(c) \equiv \varphi_\ell \vee \varphi_r$, then a number of disequation that can be further reduced are introduced (those of the form $t_i \neq s_j$). The other disequation are already in explicit solved form and are not sources of future computations.[7] we consider only φ_ℓ. φ_r will use exactly the same disequations of φ_ℓ, i.e. every disequation of the form $s_j \neq t_i$ in φ_r occurs in φ_ℓ in the form $t_i \neq s_j$ and therefore we can adopt a structure-sharing technique to avoid to reconsider this disequations.

$$\begin{array}{rcl}\varphi_\ell & \equiv & \bigvee_{i=1}^{h} \bigwedge_{j=1}^{k} (t_i \neq s_j) \ \wedge \ \bigwedge_{j=1}^{n} (Y_j \neq Y_j \cup t_i) \vee \\ & & \bigvee_{i=1}^{m} \bigwedge_{J \subseteq \{1,\ldots,k\}} (X_i \cup Y_1 \cup \ldots \cup Y_m \neq Y_1 \cup \ldots \cup Y_m \cup \bigcup_{r \in J} s_r)\end{array}$$

It is sufficient to extend the proof of Prop. 5.3 by analyzing the rewriting rule $(R3)$ defined as

$(R3)$ for each not marked node μ labeled $t_1 \cup \cdots \cup t_h \neq s_1 \cup \cdots \cup s_k$ generate $h \cdot k$ sons labeled $t_i \neq s_j$, $i = 1, \ldots, h, j = 1, \ldots, k$. Mark the node μ.

By inductive hypothesis, the trees with root labeled $t_i \neq s_j$ have a number of nodes $n_{i,j} \leq |t_i||s_j|$. Thus, the number of nodes of the tree with root labeled $t_1 \cup \cdots \cup t_h \neq s_1 \cup \cdots \cup s_k$ has $m = \sum_{i,j}^{h,k} n_{i,j} + 1$ nodes. By inductive hypothesis, $m \leq \sum_{i=1,j=1}^{h,k} |t_i||s_j| + 1 \leq (\sum_{i=1}^{h} |t_i| + h - 1)(\sum_{j=1}^{k}(|s_j| + k - 1)$, since $h > 1$ and $k > 1$. □

Observe that $\nu(c)$ generates an exponential number of disequations. So, even if the number of steps is polynomial, the real complexity of the algorithm is exponential: $O(n^3 2^n)$.

Example 5.8 *Let us consider the constraint $f(X \cup a \cup g(Y), a) \neq f(X \cup Y \cup b, X)$ and assume that $\#(X) < \#(Y) < \#(a) < \#(b) < \#(g)$. The constraint is in implicit solved form; the corresponding disjunction of constraints in intermediate solved form is*

$$X \neq a \vee X \cup a \cup g(Y) \neq X \cup Y \cup b$$

and the corresponding disjunction of constraints in explicit solved form is

$$\begin{array}{lll} X \neq a \ \vee & X \neq X \cup a \wedge Y \neq Y \cup a \ \vee & X \neq X \cup g(Y) \wedge Y \neq Y \cup g(Y) \ \vee \\ X \neq X \cup b \ \vee & \multicolumn{2}{l}{X \neq X \cup Y \wedge X \cup Y \neq X \cup a \wedge X \cup Y \neq X \cup g(Y) \wedge X \cup Y \neq X \cup a \cup g(Y)} \end{array}$$

Observe that the components of the explicit normal form precisely identify the set of possible solutions. For instance, the second disjunct forces X and Y to be not of the form $a \cup s$ for any term s.

[7] Observe that the latter equations, although inactive, are in exponential number!

6 Discussion and Related Work

In this section we present some observations regarding how our results extend to the cases of Set-constraints (Sect. 6.1) and nested sets (Sect. 6.2). Sect. 6.3 compares our results with similar proposals presented in the literature.

6.1 Set-constraints Structures

Set-constraints [2, 22] are conjunctions of literals based on the *constant symbols* $\underline{\mathbf{0}}, \underline{\mathbf{1}}$, and a non-empty set of first-order constant symbols (e.g., a, b); *function symbols:* $\cup, \cap, \complement$, where $\complement$ stands for the complement operator, and, possibly, an additional set of free function symbols (e.g., f, g). If a function symbol f is available, then also its projection f_i^{-1} on the ith argument can be used as term-constructors.

The domain on which satisfiability must be checked is the powerset of $T(\Sigma)$, with Σ containing all symbols above but $\underline{\mathbf{0}}, \underline{\mathbf{1}}, \cup, \cap, \complement, f_i^{-1}$. Moreover, the interpretation for the functions and their projection is fixed:

$$\begin{array}{rcll} (f(a_1,\ldots,a_n))^{\mathcal{P}} & = & \{f(t_1,\ldots,t_n) \,:\, \bigwedge_{i=1}^n t_i \in a_i^{\mathcal{P}}\} & \text{for } f \in \Sigma \\ (f_i^{-1}(a))^{\mathcal{P}} & = & \{t_i \,:\, \exists t_1,\ldots,t_n\, f(t_1,\ldots,t_n) \in (a)^{\mathcal{P}}\} & \text{for } f \in \Sigma \end{array}$$

$(\cdot)^{\mathcal{P}}$ is extended to set-based operators in the intuitive way. In particular the interpretation is a model of $\cup$.

Thus, the interpretation domain must be a join-semilattice with bottom. If we forbid the use of symbols $\cap, \underline{\mathbf{1}}, \complement$, and the projection symbols, we have another domain for checking $ACI1$ constraints. Of course, this model is also a model of other axioms:

$$\begin{array}{lll} (F_1^s) & f(X_1,\ldots,X_n) = f(Y_1,\ldots,Y_n) \wedge (\bigwedge_{i=1}^n X_i \neq \underline{\mathbf{0}} \vee \bigwedge_{i=1}^n Y_i \neq \underline{\mathbf{0}}) \rightarrow & f \in \Sigma \\ & \quad X_1 = Y_1 \wedge \cdots \wedge X_n = Y_n & \\ (F_2^s) & f(X_1,\ldots,X_m) = g(Y_1,\ldots,Y_n) \rightarrow (\bigvee_{i=1}^m X_i = \underline{\mathbf{0}} \wedge \bigvee_{i=1}^n Y_i = \underline{\mathbf{0}}) & f \not\equiv g, f, g \in \Sigma \\ (0) & \underline{\mathbf{0}} = f(X_1,\ldots,X_{i-1},\underline{\mathbf{0}},X_{i+1},\ldots,X_n) & f \in \Sigma, ar(f) = n, i \in \{1,\ldots,n\} \end{array}$$

Consider $T(\Sigma \cup \{\underline{\mathbf{0}}, \cup\})$. Axioms (F_1^s) and (F_2^s) imply that $s \neq t$ for all $s, t \in T(\Sigma), s \not\equiv t$.

Observe that from (0) (drawn from [22]) we get that $f(\ldots,\underline{\mathbf{0}},\ldots) = g(\ldots,\underline{\mathbf{0}},\ldots)$, for any $f, g \in \Sigma$. Thus, this model is not a model of our (F_2) axiom (when the arguments are not atoms).

In this context, one can use the set-constraints satisfiability algorithms (see, e.g., [8]) to verify satisfiability of the $ACI1$ constraint, where $s = t$ is replaced by $s \subseteq t \wedge t \subseteq s$ and $s \neq t$ is replaced by $(s \cap \complement(t)) \cup (t \cap \complement(s)) \not\subseteq \underline{\mathbf{0}}$.

6.2 Nested sets

The algorithms in this paper, as well as any other algorithm for dealing with $ACI1$ constraints when Σ contains at least a function symbol f of arity greater than 0, can be used to solve constraints on hereditarily finite and hybrid sets (where hybrid means that also terms built using "free" symbols—atoms—can be used as elements of sets). As a matter of fact, using a unary function symbols, say $\{\cdot\}$, as done in Example 4.6, sets with any level of nesting can be written. Thus, we can encode problems such as:

$$\{X, \{X, \emptyset, \{Y\}\} \cup Z \cup W\} \neq \{Z, \{\{\{X \cup Y\}, \emptyset\}\}\} \cup W$$

Set inclusion can be easily encoded by $X \subseteq Y \leftrightarrow Y = X \cup Y$. Moreover, with nested sets, membership becomes very important. This operation can be encoded as: $X \in Y \leftrightarrow Y = \{X\} \cup Y$.

Constraints of this form are special cases of formulae of *multi-level syllogistic*, whose decision problem has been studied in [7].

6.3 Related Works

In [11] Comon studies the problem of determining adequate solved forms for disunification problems in the context of quotient algebras $T(\Sigma)/=_E$ for various classes of equational theories E. Comon identifies a class of formulae, called *compact formulae*. All the three solved forms presented in this paper fulfill the requirement of being compact formulae. Additionally, Comon proves that *compact* equational theories, i.e., theories for which:

- E-unification is finitary and decidable
- each equation $s = t$, $vars(s,t) = \{X\}$ and such that $s \neq_E t$ admits a finite number of solutions in $\mathcal{H}$

guarantee that every compact formula distinct from false is satisfiable in $T(\Sigma)/=_E$ [11].

However, $ACI1$ is not compact, since equations of the type $X = X \cup a$ do not admit a finite set of solutions if $T(\Sigma)/=_{ACI1}$ is infinite—which is always the case when Σ is general. Nevertheless, we have demonstrated in the previous sections how to reduce an arbitrary disunification problem to a compact form, as well as the fact that the specific compact forms considered in our context are always satisfiable (Corollary 5.1). Thus, $ACI1$ represents a good example to indicate that compactness of the equational theory is a sufficient but *not necessary* condition for the satisfiability of formulae in compact form.

Bückert [6] introduces a general scheme for solving disunification problems in the context of an arbitrary equational theory E. Solutions of disunification problems are described through the use of *substitutions with exceptions*, i.e., entities of the form σ, Ψ where σ is a substitution and Ψ a set of substitutions. An actual solution to the disunification problem is represented by any instantiation of σ which is not an instantiation of any of the substitutions in Ψ. In the context of a theory E which is finitary w.r.t. unification, the set of all solutions to a disunification problem can be represented using a finite set of substitutions with exceptions—and additionally the Ψ component of each of them is guaranteed to be also finite. Substitutions with exceptions can be obtained from the solutions of a set of unification problems.

Nevertheless, this approach is not suitable to be used in the context of a CLP language. Each substitution with exceptions is equivalent to a formula of the type:

$$\exists \bar{W} \forall \bar{Y} (X_1 = t_1 \wedge \ldots \wedge X_n = t_n \wedge W_1 \neq s_1 \wedge \ldots \wedge W_m \neq s_m)$$

where $vars(s_1, \ldots, s_m) = \bar{Y}$, $vars(t_1, \ldots, t_n) \cap \bar{Y} = \emptyset$ and $W_1, \ldots, W_m \in \bar{W}$. This leads to the generation of formulae with arbitrary quantifications, that are inadequate to a CLP framework. Moreover, to guarantee the existence of solutions it is necessary to verify whether σ is an E-instance of any substitution in Ψ—as stems from the *Inconsistency Lemma* in [6]. This requires solving additional E-unification problems as well as having an *explicit* representation of Ψ.

In [4] Baader and Schulz develop a general technique capable of combining the satisfiability algorithms (based on substitutions with exceptions) for disjoint equational theories. The approach is general and can be applied to $ACI1$ as well. However, it provides a unique normal form that can be reached in exponential time. The solution of [4] introduces a great variety of new variables and opens a large number of alternatives. In particular, with this method one has to guess: the partition of the m variables present in the problem ($m \leq n$) into equivalence classes, a *linear ordering* over the variables (among the possible $m!$), and a *type information* for each variable, specifying to which theory E_0, E_1 the variable belongs to (2^m possible choices). This leads to an overall complexity—modulo the usual combinatorial approximations—of $\approx \sqrt{2}\left(n^{\frac{3n-2}{2}}\right)\left(\left(\frac{2}{e}\right)^{\frac{n-2}{2}}\right)$. Thus, in a particular case as that presented in this paper, it is reasonable to improve their constraint solver, developed for a universal framework. We do not introduce new variables and, using the implicit normal form we do not introduce disjunctions. Starting from a constraint made of disequations, the complexity of our approach seems to be more promising (e.g., $O(n^2)$ for the implicit normal form) and practical.

Example 6.1 *Let $E_1 = ACI1$ on $\Sigma_1 = \{\emptyset, \cup\}$ and $E_2 = f(X) = f(X) \wedge c = c$ on $\Sigma_2 = \{c, f\}$, and consider $c:\ X \cup f(Y) \neq V \cup f(Z)$. To apply the algorithm of [4], first transform c into the equivalent constraint: $A = f(Y) \wedge B = f(Z) \wedge X \cup A \neq V \cup B$, where A and B are new variables. Then consider all the partitions of the set $\{A, B, X, Y, V, Z\}$ of variables. Variables in the same partition are forced to be equal; however, not all the partitions lead to a solution of the problem. For instance, the partition*

$$\{\{X, V\}, \{Z, Y\}, \{A, B\}\}$$

can not lead to a solution, independently from the other non-deterministic choices.[8]

[8] A solution derived from this partition would be a solution of: $A = f(X) \wedge A = f(Z) \wedge X \cup A \neq X \cup A$, which is clearly unsatisfiable.

Our proposal, on the contrary, first applies the function ρ to the constraint c, obtaining the same constraint. This constraint is already in intermediate normal form, and thus it is satisfiable in every model of $ACI1F_1F_2F_3D_f$, thanks to Theorem 4.17. On demand, we can transform it in the explicit form:

$$\begin{array}{lcll} (Y \neq Z \wedge V \neq V \cup f(Y)) & \vee & (X \cup V \neq V \wedge X \cup V \neq V \cup f(Z)) & \vee \\ (Z \neq Y \wedge X \neq X \cup f(Z)) & \vee & (V \cup X \neq X \wedge V \cup X \neq X \cup f(Y)) & \end{array}$$

in which the 4 *disjuncts are all satisfiable.*

In the context of CLP with sets, three major proposals have been presented in the literature. In [18], Gervet presents a language, called *Conjunto* which incorporates a constraint solver over boolean lattices built from (flat) set intervals. The constraints can be more complex (e.g., boolean constraints) than those considered in this context, but the domain is less general. In particular, the simulation of nested sets (see Sect. 6.2) is not possible—which prevents the direct encoding of many interesting problems. Conjunto has been embedded in the recent releases of the ECLIPSE system.

In CLPS [23] the authors use a normal form similar to the implicit one presented in this paper. On the other hand, their constraint solving mechanisms appear to be based on reducing the problem to standard forward-checking and lookahead techniques. The limited literature on the topic prevents us from a deeper comparison with the capabilities of CLPS.

{log} [13, 14] is a constraint logic programming language over hybrid and hereditarily finite sets. Sets in {log} are represented using a more restricted construction—based on the use of the constant $\emptyset$ and the binary function symbol *with*. In particular, $\cup$ is not directly expressible but is derived through the use of recursive {log} definitions and from the axiomatization of *with*. Unification in this context is still NP-complete and can be seen as an instance of the cases analyzed in this paper. Disunification is relatively simpler: the constraint solvers developed for {log} [13] are capable of handling both equalities and disequalities, leading to a solved form containing only primitive constraints of the form $X = t$ and $Y \neq s$, where X, Y are variables, X occurs only once in the resulting constraint, and Y does not appear in s.

We have investigated the possibility of transforming our constraints in explicit solved form (Section 5.3) in the same solved form as the one used by {log}. This task requires the ability of forcing variables to assume only atoms as values, something which cannot be explicitly expressed in the context of a possibly unbound signature. This can be achieved through the introduction of an additional type of constraint: `kernel` (see also [13]) which is satisfied by all the atoms of the lattice used for the interpretation of the ACI1 theory.

7 Conclusion

In this paper we have studied the problem of verifying the satisfiability of conjunctions of equations and disequations w.r.t. an $ACI1$ theory. The ability to efficiently verify the satisfiability of this class of formulae is vital to the development of more general and effective CLP languages embedding sets. Existing results in the area of E-disunification (e.g., [11, 6, 4]) present general techniques which are either inadequate to the needs of a CLP framework (e.g., [6, 4]) or unsuitable to the characteristics of $ACI1$ equational theories (e.g., [11]).

The contributions of this paper can be summarized as follows:

- we have characterized the classes of structures which are suitable to model $ACI1$-like theories
- we have provided complexity results for the problem of verifying satisfiability of elementary disunification and disunification with constants
- in the general disunification case, we have characterized the axiomatization which captures the desired properties, and which corresponds to the "standard" $T(\Sigma)/=_{ACI1}$ model
- we have proposed three solved forms, increasingly more precise in the characterization of the solutions set, and developed algorithms to compute the equivalent solved form for arbitrary conjunctions of disequations. Each normal form can be trivially tested for satisfiability. Furthermore two of the three normal forms can be computed and tested in polynomial time

- we have related our results to the issues of set-constraint structures

As future work, we will continue exploring the issue of solved forms for $ACI1$ constraints, e.g., to achieve even more precise representation of the solutions set. We will also explore the use of structure sharing as a term representation technique to contain the size explosion during the unification phase of constraint solving.

Acknowledgments

The authors wish to thank Roberto Giacobazzi and Desh Ranjan for the help on this paper. A. Dovier and C. Piazza are partially supported by MURST project: *Tecniche formali per la specifica, l'analisi, la verifica, la sintesi e la trasformazione di sistemi software*. E. Pontelli is partially supported by NSF grants CDA9729848, EIA9810732, CCR9875279, and by a grant from US-Spain Research Program.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[2] A. Aiken. Set Constraints: Results, Applications and Future Directions. Technical report, University of California, Berkeley, 1994.

[3] F. Baader and W. Büttner. Unification in Commutative and Idempotent Monoids. *Theoretical Computer Science*, 56:345–352, 1988.

[4] F. Baader and K. U. Schulz. Combination Techniques and Decision Problems for Disunification. *Theoretical Computer Science*, 142:229–255, 1995.

[5] F. Baader and K. U. Schulz. Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures. *Journal of Symbolic Computation*, 21:211–243, 1996.

[6] H.-J. Bückert. Solving Disequations in Equational Theories. In E. L. Lusk and R. A. Overbeek, editors, *CADE 1988*, volume 310 of *Lecture Notes in Computer Science*, pages 517–526. Springer-Verlag, Berlin, 1988.

[7] D. Cantone, A. Ferro, and E. G. Omodeo. *Computable Set Theory, Vol. 1.* International Series of Monographs on Computer Science. Clarendon Press, Oxford, 1989.

[8] W. Charatonik and L. Pacholski. Negative Set Constraints with Equality. In *Proc. 9th Symp. Logic in Computer Science*. IEEE, 1994.

[9] K. L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–321. Plenum Press, 1978.

[10] M. Codish and V. Lagoon. Type Dependencies for Logic Programs using ACI-unification. Proc. of the 1996 Israeli Symposium on Theory of Computing and Systems, IEEE Press, pp. 136-145, 1996.

[11] H. Comon. Disunification: a Survey. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson.* The MIT Press, Cambridge, Mass., 1991.

[12] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, Mass., 1990.

[13] A. Dovier, E. G. Omodeo, E. Pontelli, and G. Rossi. {log}: A Language for Programming in Logic with Finite Sets. *Journal of Logic Programming*, 28(1):1–44, 1996.

[14] A. Dovier, C. Piazza, E. Pontelli, and G. Rossi. On the Representation and Management of Finite Sets in CLP-languages. In *Proc. Int'l Conf. and Symposium on Logic Programming*, pages 40–54. The MIT Press, 1998.

[15] A. Dovier, E. Pontelli, and G. Rossi. Set Unification Revisited. NMSU-CSTR-9817, Dept. of Computer Science, New Mexico State University, USA, October 1998.

[16] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press Inc., 1972.

[17] C. Fidge, I. Hayes, A. Martin, and A. Wabenhorst. A Set-theoretic Model for Real-Time Specification and Reasoning. In *Mathematics of Program Construction*, Springer Verlag, 1998.

[18] C. Gervet. Interval Propagation to Reason about Sets: Definition and Implementation of a Practical Language. *Constraints*, 1:191–246, 1997.

[19] G. Grätzer. *General Lattice Theory.* Birkhäuser Verlag Basel und Stuttgart, 1978.

[20] J. Jaffar, M. Maher, K. Marriot, and P. Stuckey. The Semantics of Constraint Logic Programs. *Journal of Logic Programming*, 37:1–46, 1998.

[21] D. Kapur and P. Narendran. NP-Completeness of the Set Unification and Matching Problems, In J. H. Siekmann ed., 8th International Conference on Automated Deduction, LNCS n. 230, pp. 489–495, Springer-Verlag, 1986.

[22] D. Kozen. Logical Aspects of Set Constraints. In *Procs. Conf. on Computer Science Logic*, Vol. 832 LNCS, pp. 175–188. Springer-Verlag, 1993.

[23] B. Legeard and E. Legros. Short Overview of the CLPS System. In *Proc. Third Int'l Symp. on Programming Language Implementation and Logic Programming*, Vol. 528 LNCS, pp. 431–433. Springer-Verlag, 1991.

[24] M. J. Maher. Complete Axiomatizations of the Algebras of Finite, Rational and Infinite Trees. In *Proceedings of* 3^{rd} *Symposium Logic in Computer Science* (1988), 349–357.

[25] A. Robinson. *Introduction to Model Theory and to the Metamathematics of Algebra.* North Holland, Amsterdam, 1963.

[26] J. H. Siekmann. Unification Theory. In C. Kirchner, editor, *Unification.* Academic Press, 1990.