Intelligent Information Systems
Proceedings of the Workshop held in Wigry, Poland
Polish Academy of Sciences, 1994, ISBN 83-900820-3-9, pp. 25-39.

TRUE EQUALITY CONSTRAINTS OVER AN OPEN UNIVERSE OF TERMS[1]

JAN A. PLAZA

University of Miami
Department of Mathematics and Computer Science
P.O. Box 249 085
Coral Gables, Florida 33124
U.S.A.
janplaza@math.miami.edu

Abstract. For a first-order language $\mathcal{L}$, the Herbrand universe consists of ground terms of $\mathcal{L}$. We defined [15, 16, 17] an *ω-Herbrand universe* for $\mathcal{L}$ by introducing the following modification: first add to $\mathcal{L}$ countably many new individual constants, and then form the set of all ground terms of the resulting language. Newly added constants make this universe "open" and suitable for modeling those situations in databases where not all entities are known in advance.

In this research we investigate the theory of equality determined by ω-Herbrand interpretations, we give its axiomatization, and prove that it is complete and decidable. For a formula $\exists_x B$, the decision algorithm not only tells whether this formula is a consequence of our equality theory, but also computes substitutions θ such that $B\theta$ is true in all ω-Herbrand interpretations.

To put these results in a broader context, let us mention that algorithm above can be used to generate answer substitutions in a logic programming language which admits as constraints arbitrary first-order formulas built from equalities of terms. Unlike in currently implemented logic programming languages, such constraint solving mechanism guarantees that equality is understood as in the first order logic, and treated in a fully declarative way. This approach has a potential for being combined with the others, as well as a potential for parallelization.

Area: Foundations of Logic Programming.

Keywords: constraint logic programming, Herbrand models, equality, completeness, decidability.

1 INTRODUCTION

In this section we will give informal examples which explain the notions of an "open universe" and "true negation". We will also briefly relate problems considered in this paper to other research.

Consider program P containing only the following two facts: $p(a)$ and $p(b)$. In every Herbrand model M for P (in fact there is only one such model) we have $M \models \forall_x p(x)$. The conventional Herbrand universe contains no elements besides a and b: the universe is "closed". We will see that ω-Herbrand models behave differently: in every ω-Herbrand model M for P we have $M \models p(a)$ and $M \models p(b)$, but there exists an ω-Herbrand model M_0 for P, such that $M_0 \models \neg\forall_x p(x)$. The ω-Herbrand universe contains elements other than a and b: the universe is "open". Programming over an open universe is suitable for situations involving a growing knowledge base, where at any given moment we do not want to exclude from considerations objects whose descriptions will be incorporated later, and where "for all" means "for all entities which will be incorporated into the database".

Prolog, Prolog II, Prolog III [4, 5, 6, 7] compute over an universe of finite or infinite terms (finite or rational trees), but the implemented comparison operations are based on unifiability and they differ from "true equality". Consider the following logic program

$$p(x,y) \leftarrow \neg x{=}y \wedge q(x,y)$$
$$q(a,a) \leftarrow true$$
$$q(a,b) \leftarrow true$$

where x, y are variables and a, b are constants. With the query $p(x,y)$, it would be natural to expect $x{=}a$, $y{=}b$ as a unique answer. But this does not happen in Prolog. In Prolog there are two ways of understanding equality.

Prolog's $t_1{=}t_2$ means that terms t_1, t_2 are unifiable. Therefore Prolog's program

```
p(X,Y) :- not X=Y, q(X,Y).
q(a,a) :- true.
q(a,b) :- true.
```

with the query $p(X,Y)$, responds that there are no solutions.

Prolog's $t_1 == t_2$ means that terms t_1, t_2 are syntactically identical. Therefore Prolog's program

```
p(X,Y) :- not X==Y, q(X,Y).
q(a,a) :- true.
q(a,b) :- true.
```

with the query $p(X,Y)$, returns two sets of answers: `X=a,Y=a`, and `X=a,Y=b`.

Both forms of equality in Prolog differ from "true equality". Of course it would be possible to use some non-logical features of the language (change the order of literals in a clause or delay selection of a literal) but we will propose a different solution.

In this paper we present a formal theory of equality CET^{ω} which can be used as a basis for implementing a logic programming language that handles equality according to our initial expectations. We will see that it will be possible to handle programs much more complex than that of the example above: arbitrary constraints involving equality, negation and quantifiers can be used in clauses; for instance:

$$p(x) \leftarrow \forall_y (\neg x{=}f(y)) \wedge q(x)$$

The idea of handling such programs is the following: perform a generalized SLD-resolution, selecting only those subgoals which do not contain the equality predicate. When in this way you obtain a goal with no predicates except =, apply a supplementary algorithm which will compute expected answer substitutions.

In our initial example the resolution would end with:

$$\neg x{=}y \wedge ((x{=}a \wedge y{=}a) \vee (x{=}a \wedge y{=}b))$$

In our equality theory:

$$CET^{\omega} \vdash \neg x{=}y \wedge ((x{=}a \wedge y{=}a) \vee (x{=}a \wedge y{=}b)) \quad \rightarrow \quad x{=}a \wedge y{=}b$$

and the supplementary algorithm, which is related to the decision procedure for CET^{ω}, would return the desired answer: $x{=}a,\ y{=}b$.

In this paper we concentrate on the issues of mathematical logic related to our equality theory. We discuss main results and sketch ideas of the proofs. Full proofs can be found in the authors doctoral dissertation [15], they will be also included in the full version of the paper. A design for a logic programming language which incorporates CET^{ω} has been presented in [17].

This work joins elements of the theory of logic programming, mathematical logic and algebraic semantics. The reader who seeks detailed expositions of these subjects is referred to [13, ?, ?].

The notion of an ω-Herbrand universe – an open version of the universe of terms has been defined and investigated in [15, 16]. As shown there, for a wide class of programs, the corresponding operators on lattices of ω-Herbrand interpretations reach their least fixed points in ω steps. This makes the ω-Herbrand universe a feasible domain for logic programming.

In [8], Jaffar and Lassez proposed a theory which covers a variety of constraint logic programming languages. The domains of constraints are required to be satisfaction-complete and solution-compact. As our domain is not solution-compact, it is not covered by that theory, and requires a separate treatment.

In [11, 12] Kunen proved that if $\mathcal{L}$ contains infinitely many function symbols then Clark's Equality Theory CET is a complete and decidable description of the Herbrand universe $U_{\mathcal{L}}$. In this paper we consider an "open" counterpart of $U_{\mathcal{L}}$ and construct a complete and decidable description in a language with a finite number of function symbols.

Mancarella, Martini and Pedreschi proved in [14] that CET augmented with the Closed Domain Axiom describes $U_{\mathcal{L}}$ in a complete way; unlike this description, ours will be within the first order logic, and will concern an open universe.

Acknowledgments. Warmest thanks to Professor Helena Rasiowa and to Professor Melvin Fitting.

2 ω-HERBRAND INTERPRETATIONS

In this section we propose an alternative version of the notion of Herbrand interpretations. The formal theory of equality determined by such interpretations will be investigated throughout the paper.

Definition 1.

If a first order language language $\mathcal{L}$ contains no individual constants, let $\mathcal{L}'$ be an extension of $\mathcal{L}$ obtained by adding one individual constant; if in $\mathcal{L}$ there is an individual constant, let $\mathcal{L}'=\mathcal{L}$. By an *Herbrand universe* $U_{\mathcal{L}}$ for $\mathcal{L}$ we understand the set of all ground terms of $\mathcal{L}'$.

By a *Herbrand Interpretation* for $\mathcal{L}$ we understand any interpretation M for $\mathcal{L}$ whose universe is $U_{\mathcal{L}}$ and in which function symbols are interpreted in the natural way: $f^M(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$.

By an *Herbrand model* for a set Γ of formulas we understand an Herbrand interpretation which is a model for Γ.

The notion of a Herbrand interpretation is an basic tool in the theory of logic programming. We propose the following alternative which overcomes some deficiencies of the original setting.

Definition 2 (ω-Herbrand Interpretations).

Throughout this paper by a *language* we understand a countable first-order language containing finitely many function symbols. (Individual constants are considered 0-ary function symbols.)

Let $\mathcal{L}$ be a language, and let $\mathcal{L}^{\mathcal{K}}$ result by adding to the alphabet of $\mathcal{L}$ a countable set $\mathcal{K} = \{k_i \mid i < \omega\}$ of new individual constants.

By the *ω-Herbrand universe* $U^{\omega}_{\mathcal{L}}$ for $\mathcal{L}$ we understand the set of all ground terms of the language $\mathcal{L}^{\mathcal{K}}$. Members of the set $\mathcal{K}$ will be called *free constants.*

By a *ω-Herbrand interpretation* for $\mathcal{L}$ we understand any interpretation M for $\mathcal{L}$ whose universe is $U^{\omega}_{\mathcal{L}}$, and in which function symbols are interpreted in the natural way: $f^M(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$.

We will use the symbol $U^{\omega}_{\mathcal{L}}$ to denote not only the set of all ground terms of $\mathcal{L}^{\mathcal{K}}$, but also to denote the ω-Herbrand interpretation with universe $U^{\omega}_{\mathcal{L}}$, in which $=$ is interpreted as equality, and all the remaining predicates are interpreted as $\emptyset$. The meaning will be clear from the context.

By an *ω-Herbrand model* for a set Γ of formulas we understand an ω-Herbrand interpretation which is a model for Γ.

In the remaining part of this section we will compare some features of the two versions of Herbrand interpretations.

Theorem 3 (Completeness resp. to ω-Herbrand Interpretations).
Let $\Gamma \cup \{B\}$ be a set of formulas in a language $\mathcal{L}$ without equality. Then, $\Gamma \not\models B$ implies that there exists an ω-Herbrand interpretation I such that $I \models \Gamma$ and $I \not\models B$.

The theorems in this and in the next section are proved using methods of algebraic semantics. For first-order logics such methods have been developed in 1950's by H. Rasiowa and R. Sikorski. The algebraic approach suits our needs especially well, because it turns out that universes of so called "canonical algebraic models" are exactly ω-Herbrand universes. As algebraic semantics is a well investigated area of mathematical logic, one can expect that applying its ideas to ω-Herbrand interpretations may quickly bring new interesting results.

Remark. Let $\mathcal{L}$ be a language without equality. It is known that:

If P is a set of Horn clauses in $\mathcal{L}$ then:
P is unsatisfiable iff P has no Herbrand model.

The straightforward counter-example with P being $\{p(c), \exists_x(\neg p(x))\}$ shows that in this proposition one cannot drop the assumption that formulas in P are Horn clauses, cf. [13] p. 17. But if we consider our alternative to Herbrand models we can drop this assumption (cf. [15, 16]):

If P is any set of formulas in $\mathcal{L}$ then:
P is unsatisfiable iff P has no ω-Herbrand model.

Remark. Consider a language $\mathcal{L}$ without equality. For Horn programs P and ground atomic formulas A, the *Herbrand rule* (cf. [13] p. 100) is defined as:

If $Comp(P) \cup \{A\}$ has no Herbrand model, infer $\neg A$.

This rule is strictly weaker than Reiter's Closed World Assumption Rule (CWA). With ω-Herbrand interpretations we may define an *ω-Herbrand rule*:

If $Comp(P) \cup \{A\}$ has no ω-Herbrand model, infer $\neg A$.

Now, by the theorem above, for Horn programs and ground atomic formulas, the ω-Herbrand rule coincides with CWA.

Remark. Another major difference between conventional Herbrand interpretations and ω-Herbrand interpretations is demonstrated by the behavior of operators associated with logic programs. Let P be a program in language $\mathcal{L}$ consisting of formulas $A \leftarrow B$ where B is arbitrary positive formula (with $\forall$, $\exists$). Let T_P and T_P^ω be operators associated with P, acting respectively on the lattices of all conventional Herbrand interpretations and all ω-Herbrand interpretations. Because of the presence of general quantifiers, these operators need not be continuous. In general T_P requires more than ω steps from the bottom of the lattice to reach its least fix-point; on the other hand T_P^ω always reaches its least fix-point in ω steps. This issue is discussed in [15, 16].

3 THEORY OF EQUALITY

In this section we introduce equality theory $CET_{\mathcal{L}}^{\omega}$. The theory guarantees not only that $=$ is a congruence, but also that function symbols are interpreted in a "free" way and that "free constants" exist in its models. Later we will show that $CET_{\mathcal{L}}^{\omega}$ is a complete and decidable description of the ω-Herbrand universe $U_{\mathcal{L}}^{\omega}$.

Definition 4. Let $\mathcal{L}$ be a language with equality. Clark's Equality Theory $CET_{\mathcal{L}}$ is based on the following axioms:

1. $x{=}x$
2. $x{=}x' \rightarrow x'{=}x$
3. $x{=}x' \wedge x'{=}x'' \rightarrow x{=}x''$
4. $x_1{=}x'_1 \wedge \ldots \wedge x_n{=}x'_n \rightarrow f(x_1, \ldots, x_n){=}f(x'_1, \ldots, x'_n)$
   for any function symbol f.
5. $x_1{=}x'_1 \wedge \ldots \wedge x_n{=}x'_n \wedge p(x_1, \ldots, x_n) \rightarrow p(x'_1, \ldots, x'_n)$
   for any predicate symbol p.
6. $f(x_1, \ldots, x_n) {\neq} g(x'_1, \ldots, x'_m)$

for any pair of different function symbols f and g.

7. $f(x_1,\ldots,x_n)=f(x'_1,\ldots,x'_n) \rightarrow x_1=x'_1 \wedge \ldots \wedge x_n=x'_n$
for any function symbol f.

8. $t\neq x$ for any term t containing x, but not identical with x.

Axioms 1-5 state that $=$ is a congruence in models, and this is what is always assumed in first-order theories with equality. Axioms 6-8 assure that function symbols are interpreted in models in a "free" way. Axiom 8 is related to the occurs check in the term unification algorithm. Notice that the axioms are not independent: 1 and 5 imply 2 and 3. Using semantical methods one can show that 8 is not a consequence of 1-7.

Our main task in this section is to give a syntactical description of $U^\omega_\mathcal{L}$. As $U^\omega_\mathcal{L} \models CET_\mathcal{L}$, the theory $CET_\mathcal{L}$ is a correct description of $U^\omega_\mathcal{L}$. However $CET_\mathcal{L}$ is not complete, even if the language $\mathcal{L}$ contains no predicate symbols except $=$. (To see that $CET_\mathcal{L}$ is not complete consider the language with one individual constant c, the predicate symbol $=$, and with no other symbols; using semantics, observe that neither $CET_\mathcal{L} \vdash \exists_x(x\neq c)$ nor $CET_\mathcal{L} \vdash \neg\exists_x(x\neq c)$.) For languages containing no predicate symbols except $=$ we will seek a complete description of $U^\omega_\mathcal{L}$. To obtain such a description we have to add to $CET_\mathcal{L}$ axioms guaranteeing that in any model there exist infinitely many elements that are neither interpretations of individual constants nor values of functions. This is done in the next definition.

Definition 5 (Equality Theory $CET^\omega_\mathcal{L}$).
Let $\mathcal{L}$ be a language with equality. By $CET^\omega_\mathcal{L}$ we understand the extension of $CET_\mathcal{L}$ obtained by adjoining the following axioms, for every $n<\omega$:

$$\exists_{x_1,\ldots,x_n} \bigwedge_{n\geq i\neq j\leq n} x_i\neq x_j \wedge \bigwedge_{i\leq n} \bigwedge_{f\in Func} \neg is_f(x_i)$$

where $is_f(x)$ is an abbreviation for $\exists_{y_1,\ldots,y_m}(x=f(y_1,\ldots,y_m))$,
(for a 0-ary function symbol c, i.e. for an individual constant, $is_c(x)$ means $x=c$).

Notice that $CET^\omega_\mathcal{L}$ can be defined only for languages containing finitely many function symbols, otherwise the conjunction $\bigwedge_{f\in Func}\neg is_f(x_i)$ would not be finite. This is on of the differences between our approach and that of Kunen [11], (also cf. [19]).

Notice that $CET^\omega_\mathcal{L}$ does not have any finite models.

We have $U^\omega_\mathcal{L} \models CET^\omega_\mathcal{L}$ and this shows that $CET^\omega_\mathcal{L}$ is a correct description of $U^\omega_\mathcal{L}$. $CET^\omega_\mathcal{L}$ is also a more accurate description of $U^\omega_\mathcal{L}$ than $CET_\mathcal{L}$ — for instance $CET_\mathcal{L}$ does not exclude the conventional Herbrand universe $U_\mathcal{L}$ as a model, while $CET^\omega_\mathcal{L}$ does. To the problem of how well $CET^\omega_\mathcal{L}$ describes $U^\omega_\mathcal{L}$, we will come back in section 5.

Now let us recall the concept of a conservative extension of a theory.

Definition 6. Let $\mathcal{T}_1$, $\mathcal{T}_2$ be theories, respectively in languages $\mathcal{L}_1$, $\mathcal{L}_2$. Then:

1. $\mathcal{T}_2$ is called an *extension* of $\mathcal{T}_1$, if $\mathcal{L}_2$ is an extension of $\mathcal{L}_1$, and for every $B\in\mathcal{L}_1$, $\mathcal{T}_1 \models B$ implies $\mathcal{T}_2 \models B$.

2. $\mathcal{T}_2$ is called a *conservative extension* of $\mathcal{T}_1$, if $\mathcal{T}_2$ is an extension of $\mathcal{T}_1$, and for every $B \in \mathcal{L}_1$, $\mathcal{T}_2 \models B$ implies $\mathcal{T}_1 \models B$.
3. $\mathcal{T}_2$ is called a *linguistic extension* of $\mathcal{T}_1$, if $\mathcal{L}_2$ is an extension of $\mathcal{L}_1$, and the sets of axioms of $\mathcal{T}_1$, $\mathcal{T}_2$ are the same.
4. If $\mathcal{L}_2$ is obtained by adjoining a new predicate symbol p to the alphabet of $\mathcal{L}_1$, and there exists $B \in \mathcal{L}_1$ such that theory $\mathcal{T}_2$ is obtained by adjoining the formula $p(\boldsymbol{x}) \equiv B(\boldsymbol{x})$ to the axioms of $\mathcal{T}_1$, then $\mathcal{T}_2$ is called a *definitional extension* of $\mathcal{T}_1$.

It is well known that linguistic or definitional extensions are conservative.

Now we establish two basic facts concerning extensions of theories and involving $CET^{\omega}_{\mathcal{L}}$.

Proposition 7 (Extensions with Equality).

1. *Let $\mathcal{L}$ be a language without equality. Let $\mathcal{L}^{=}$ be the extension of $\mathcal{L}$ obtained by adjoining the symbol $=$ to the alphabet. Then for any theory $\mathcal{T}$ in $\mathcal{L}$, $\mathcal{T} \cup CET^{\omega}_{\mathcal{L}^{=}}$ is a conservative extension of $\mathcal{T}$.*
2. *Let $\mathcal{L}$ and $\mathcal{L}'$ be languages with equality, such that $\mathcal{L} \subseteq \mathcal{L}'$. Then for any theory $\mathcal{T}$ in $\mathcal{L}$ the theory $\mathcal{T} \cup CET^{\omega}_{\mathcal{L}'}$ is a conservative extension of $\mathcal{T} \cup CET^{\omega}_{\mathcal{L}}$.*

Thanks to the second part of this proposition it is correct to say e.g. "α_1 is equivalent in CET^{ω} to α_2" without specifying the language. Indeed, assertions $CET^{\omega}_{\mathcal{L}} \vdash \alpha_1 \equiv \alpha_2$ and $CET^{\omega}_{\mathcal{L}'} \vdash \alpha_1 \equiv \alpha_2$ are equivalent for any languages $\mathcal{L}$, $\mathcal{L}'$ containing α_1 and α_2.

4 NORMAL FORMS

In this section we consider languages $\mathcal{L}$ containing no predicate symbols except $=$. We introduce normal forms of formulas of $CET^{\omega}_{\mathcal{L}}$. Normal forms are used as the main technical tool in the proofs of decidability and completeness of this theory. Later, transformations leading to normal forms will be used in algorithms incorporated in resolution systems.

We say "every formula B in the class $\mathcal{C}$ is *effectively equivalent* in theory $\mathcal{T}$ to a formula in the class $\mathcal{C}'$", if there exists an algorithm which takes as input any $B \in \mathcal{C}$ and returns as output $B' \in \mathcal{C}'$, such that $\mathcal{T} \vdash B \equiv B'$. To motivate the notion of effective equivalence, notice that every formula in the language of arithmetic is equivalent in $Th(\mathcal{N})$ either to $\top$ or to $\bot$, but as $Th(\mathcal{N})$ is undecidable, this equivalence is not effective — there is no algorithm that could perform the task of transforming formulas in this way.

The following proposition may be used to establish in CET^{ω} normal forms of positive quantifier-free formulas.

Proposition 8. *Any equality $t'{=}t''$ of terms is effectively equivalent in CET^{ω} to $\top$, or to $\bot$, or to $\bigwedge_i x_i{=}t_i$, where x_i's are variables and t_i's are terms.*

This proposition is illustrated by the following example.

Example 1. In CET^{ω}:

1. $(f(x_1, g(x_2, c))=f(h(c), x_3)) \equiv (x_1=h(c) \wedge x_3=g(x_2, c))$.
2. $(x=x) \equiv \top$.
3. $(f(x_1, f(x_2, x_3))=f(x_2, x_1)) \equiv (x_1=x_2 \wedge x_1=f(x_2, x_3))$ and also $(f(x_1, f(x_2, x_3))=f(x_2, x_1)) \equiv \bot$.

It turns out, that in general, formulas of $CET^{\omega}_{\mathcal{L}}$ are not equivalent to quantifier-free formulas. Normal forms for arbitrary formulas will require using existential quantifiers and negated predicates is_f (which contain a hidden universal quantifier). Normal forms will be disjunctions of canonical formulas. A canonical formula with free variables $x_i : i \in I$ will express "each x_i is of the form t_i, with certain restrictions placed on t_i's". Here is the precise definition.

Definition 9 (Canonical Formulas in CET^{ω}).
By a *canonical formula* we understand the formula $\top$ or any formula

$$\exists \boldsymbol{y}(\bigwedge_{i\in I} x_i=t_i \wedge \bigwedge_{j\in J} y_j \neq t'_j \wedge \bigwedge_{\langle f,k\rangle\in K} \neg is_f(y_k))$$

where:

1. $\boldsymbol{y}=\langle y_1, \ldots, y_m\rangle$, $m \geq 0$.
2. $y_j \notin \mathrm{var}(t'_j)$ for every $j=1, \ldots, m$.
3. $\mathrm{var}(\bigwedge_{j\in J} y_j \neq t'_j \wedge \bigwedge_{\langle f,k\rangle\in K} \neg is_f(y_k)) \subseteq \bigcup_{i\in I} \mathrm{var}(t_i) = \{y_1, \ldots, y_m\}$.

(This implies that x_i's are all the free variables in the formula, and neither of x_i's occurs in t_i's or t'_j's).

By a *positive canonical formula* we understand the formula $\top$ or a canonical formula with additional requirement $J = K = \emptyset$.
$\top$

Notice that every canonical formula is satisfiable in $U^{\omega}_{\mathcal{L}}$.

Theorem 10. *Every inequality $t' \neq t''$ of terms is effectively equivalent in CET^{ω} either to $\top$, or to $\bot$, or to a disjunction of canonical formulas whose free variables are contained among the free variables of $t' \neq t''$.*

This theorem is illustrated by the following example.

Example 2. In CET^{ω}, the formula

$$f(x_1, g(x_2)) \neq f(x_2, g(f(x_3, c)))$$

is equivalent to:

$$x_1 \neq x_2 \vee x_2 \neq f(x_3, c)$$

is equivalent to:

$$\begin{aligned}&\exists_{y_1,y_2}(x_1=y_1 \wedge x_2=y_2 \wedge y_1 \neq y_2) \vee\\ &\exists_{y_1}(x_2=y_1 \wedge \neg is_f(y_1)) \vee\\ &\exists_{y_1,y_2,y_3}(x_2=f(y_1, y_2) \wedge x_3=y_3 \wedge y_1 \neq y_3) \vee\\ &\exists_{y_1,y_2}(x_2=f(y_1, y_2) \wedge \neg is_c(y_2)).\end{aligned}$$

We leave it to the reader to check that Theorem 10 does not generalize to arbitrary quantifier-free formulas. For instance formulas $x_1 \neq x_2 \wedge x_1 \neq f(x_2)$ or $x_1 \neq f(x_2) \wedge x_2 \neq f(x_1)$ cannot be represented as (finite) disjunctions of canonical formulas. The situation would change if we allowed countable disjunctions. This cannot be done in a first-order language. However in models we can consider countable unions of sets. We will come back to this idea in section 6. Now we will extend Theorem 10 to arbitrary formulas (with quantifiers).

Theorem 11 (Normal Form Theorem).
Every formula B containing no predicate symbols except $=$ is effectively equivalent in CET^{ω} either to $\top$ or to $\bot$ or to a disjunction of canonical formulas whose free variables are contained among the free variables of B. Moreover, if B is a positive formula then the formulas in the resulting disjunction are positive canonical formulas.

For the proof of this theorem we consider a linguistical extension of CET^{ω} with predicate symbols is_f added to the language. We prove that any Σ_1-formula in the extended language is equivalent either to $\top$, or to a disjunction of canonical formulas. We prove that negation of any canonical formula is equivalent to a disjunction of canonical formulas. From that, the theorem follows.

5 DECIDABILITY AND COMPLETENESS

This section completes our answer to the question of how well $CET^{\omega}_{\mathcal{L}}$ describes the ω-Herbrand universe $U^{\omega}_{\mathcal{L}}$. We prove that $CET^{\omega}_{\mathcal{L}}$ is a complete and decidable but not categorical description of $U^{\omega}_{\mathcal{L}}$. Moreover we discuss basic properties of models of $CET^{\omega}_{\mathcal{L}}$.

The following theorem is the central theorem in our investigations of $CET^{\omega}_{\mathcal{L}}$. Its proof is related to the method of elimination of quantifiers and uses the Normal Form Theorem 11. In typical elimination of quantifiers, one shows that every formula is equivalent to a Δ_0-formula. This is not true in the case of $CET^{\omega}_{\mathcal{L}}$, and instead, we show that in a certain conservative extension of $CET^{\omega}_{\mathcal{L}}$, every formula is equivalent to a Σ_1-formula.

Theorem 12 (Decidability and Completeness of $CET^{\omega}_{\mathcal{L}}$).
Let $\mathcal{L}$ be a language containing no predicate symbols except $=$. Then:

1. *$CET^{\omega}_{\mathcal{L}}$ is decidable,*
2. *$CET^{\omega}_{\mathcal{L}}$ is complete,*
3. *$CET^{\omega}_{\mathcal{L}} = Th(U^{\omega}_{\mathcal{L}})$.*

Notice that the assumption of the theorem is essential. If $\mathcal{L}$ contained predicate symbols different from $=$, then $CET^{\omega}_{\mathcal{L}}$ restricted to formulas without $=$ would coincide with the predicate calculus, hence it would be neither complete nor decidable. The next theorem will show however that if we limit our attention to positive formulas with guards, we can admit predicate symbols other than $=$.

Definition 13.
By a *positive formula* we understand any formula built from atomic formulas, including $\top$ and $\bot$, by means of $\wedge$, $\vee$, $\forall$, $\exists$.

By a *guard* we understand any formula containing no predicates except $=$.

By a *positive formula with guards* we mean any formula $B(S_1/p_1, \ldots, S_n/p_n)$ where $B(p_1, \ldots, p_n)$ is a positive formula with predicate variables $p_1, \ldots, p_n$ but does not contain $=$, and where $S_1, \ldots, S_n$ are guards.

Notice that a positive formula with guards need not be a positive formula. Positive formulas with guards may contain some, but not arbitrary, negative information.

It is also important to emphasize that formulas containing $\neg$ but equivalent to those built only by means of $\wedge$, $\vee$, $\forall$, $\exists$, are not considered positive. This is because such an equivalence is not effective: there is no algorithm which, if given a formula as input, outputs correctly an equivalent positive formula or the statement "The input formula is not equivalent to any positive formula". (This results from the fact that it is decidable whether a positive formula is a tautology, and from the undecidability of the predicate calculus.)

Theorem 14. *Let $\mathcal{L}$ be a language with equality. Then for positive formulas B with guards, it is decidable whether $CET^{\omega}_{\mathcal{L}} \vdash B$.*

Example 3. Let $\mathcal{L}$ be a language with equality. According to part 3 of Theorem 12, $CET^{\omega}_{\mathcal{L}}$ contains exactly those sentences that are true in $U^{\omega}_{\mathcal{L}}$. This implies that $U^{\omega}_{\mathcal{L}}$ is a submodel of every normal model for $CET^{\omega}_{\mathcal{L}}$. Thus one may ask whether normal models for $CET^{\omega}_{\mathcal{L}}$ can be strictly bigger than $U^{\omega}_{\mathcal{L}}$. The answer is "yes": Let $\mathcal{L}$ be a language containing one unary function symbol f and the symbol $=$. Consider an interpretation whose universe contains the ω-Herbrand universe $U^{\omega}_{\mathcal{L}}$ and countably many additional elements e_i , i being an integer (negative, 0, or positive). Let f be interpreted in $U^{\omega}_{\mathcal{L}}$ in the natural way and in the remaining part set $f(e_i){=}e_{i+1}$. The interpretation defined in this way is a model for $CET^{\omega}_{\mathcal{L}}$. This shows that the theory $CET^{\omega}_{\mathcal{L}}$ is not categorical. In general, if we consider other languages (possibly containing function symbols of greater arities), normal models for $CET^{\omega}_{\mathcal{L}}$ may contain, besides a standard $U^{\omega}_{\mathcal{L}}$ part, also "trees infinite in both directions". Notice however that axiom 8 assures us that no model for $CET^{\omega}_{\mathcal{L}}$ contains "loops" such as $e_0, e_1, \ldots, e_{n-1}$ with $f(e_i){=}e_{(i+1)\mathrm{mod}(n)}$.

6 GUARDED SUBSTITUTIONS

In this section we consider substitutions constrained by conditions involving inequations, like:

$$\{f(x_5)/x_1,\ c/x_2,\ x_4/x_3\} \text{ WHERE } (x_5{\neq}x_4 \wedge \forall_y\, x_5{\neq}f(y)).$$

We also introduce canonical forms of such substitutions; they are related to canonical formulas of $CET^{\omega}_{\mathcal{L}}$. We interpret substitutions as subsets of (a power of) the ω-Herbrand universe, and prove that under this interpretation substitutions are equivalent to finite unions of canonical substitutions.

First, terminology and notation. By a *substitution* we understand any partial function from the set $Var = \{x_i \mid i{<}\omega\}$ of variables of language $\mathcal{L}$, into the set of terms of $\mathcal{L}$. Substitutions are written as sets of elementary substitutions, $\theta = \{t_i/x_i \mid i{\in}I\}$ where I is a set of natural numbers. Any substitution $\{t_i/x_i \mid i{\in}I\}$ which is not total, is identified with the following total substitution: $\{t_i/x_i \mid i{\in}I\}{\cup}\{x_i/x_i \mid i{\in}(\omega-I)\}$.

Substitution $\theta = \{t_i/x_i \mid i \in I\}$ is called *finite* if the set $\{i \in I \mid t_i$ is distinct from $x_i\}$ is finite. The symbol ϵ stands for the *identity substitution*: $\epsilon = \{x_i/x_i \mid i < \omega\}$. Symbols such as $\theta[f(x)/x_2, c/x_4]$ denote modifications of substitutions.

Given an interpretation of a language $\mathcal{L}$, by a *valuation* we always understand a total function from the set *Var* of variables of $\mathcal{L}$, into the domain of the interpretation. However, departing from the formal style, we may sometimes specify only a finite fragment of a valuation, if the rest is irrelevant. This will be the case with the notion of satisfiability: $M \models B[\nu]$ depends only on the part of the valuation ν, that concerns variables occurring free in B. Symbols such as $\nu[e_1/x_2, e_5/x_4]$ denote modifications of valuations.

Definition 15.

1. If $\theta = \{t_i/x_i \mid i<\omega\}$ is a substitution, by $\tilde{\theta}$ we denote the sequence $\langle t_0, t_1, t_2, \ldots\rangle$.
2. If $\nu : \{x_i \mid i < \omega\} \longrightarrow |M|$ is a valuation, by $\tilde{\nu}$ we denote the sequence $\langle x_0[\nu], x_1[\nu], x_2[\nu], \ldots\rangle$ of elements of $|M|$ substituted for the consecutive variables.

Definition 16 (Canonical Guarded Substitutions).
By a *guard* we understand any formula that does not contain predicate symbols other than $=$. By a *guarded substitution* we understand any expression (θ WHERE S) such that θ is a substitution and S is a guard. By a *canonical guarded substitution* we understand any guarded substitution (θ WHERE S) such that:

1. θ is $\{t_i/x_i \mid i \in I\}$,
2. S is either $\top$ or $\bigwedge_{j \in J} y_j \neq t'_j \wedge \bigwedge_{\langle f,k\rangle \in K} \neg is_f(y_k)$,
3. $y_j \notin \mathrm{var}(t'_j)$ for every $j = 1, \ldots, m$,
4. $\mathrm{var}(S) \subseteq \bigcup_{i \in I} \mathrm{var}(t_i)$.

The reader should compare this definition with the definition 9 of canonical formulas in CET^ω.

For conventional substitutions it is defined how to compose them, and how to apply them to formulas. The next definition extends these concepts to guarded substitutions.

Definition 17.

1. The result (θ_1 WHERE S_1)(θ_2 WHERE S_2) of composing guarded substitutions (θ_1 WHERE S_1) and (θ_2 WHERE S_2) is defined as ($\theta_1\theta_2$ WHERE $(S_1\theta_2) \wedge S_2$).

2. The result B(θ WHERE S) of applying guarded substitution (θ WHERE S) to a formula B is defined as $B\theta \vee \neg S$.

Notice that if the language $\mathcal{L}$ of B does not contain the symbol $=$, then B(θ WHERE S) is a formula in the extended language $\mathcal{L}^=$.

Notice also that B(θ WHERE S) is equivalent to $S \rightarrow (B\theta)$; we did not use the other formula in the definition of B(θ WHERE S) because we wanted to achieve the following effect: If B is a positive formula with guards, so is B(θ WHERE S).

As $B(\theta \text{ WHERE } \top) \equiv B\theta$ we see that the notion of a guarded substitution generalizes conventional notion of a substitution. As $(\theta \text{ WHERE } \top)$ can be identified with just θ, all further results concerning guarded substitutions will apply to conventional substitutions as well. Notice however that, unlike conventional substitutions, guarded substitutions do not always commute with Boolean operations. We have:

$$\top(\theta \text{ WHERE } S) \equiv \top$$
$$(B_1 \wedge B_2)(\theta \text{ WHERE } S) \equiv B_1(\theta \text{ WHERE } S) \wedge B_2(\theta \text{ WHERE } S)$$
$$(B_1 \rightarrow B_2)(\theta \text{ WHERE } S) \equiv B_1(\theta \text{ WHERE } S) \rightarrow B_2(\theta \text{ WHERE } S)$$
$$\perp(\theta \text{ WHERE } S) \not\equiv \perp$$
$$(\neg B)(\theta \text{ WHERE } S) \not\equiv \neg(B(\theta \text{ WHERE } S))$$
$$(B_1 \vee B_2)(\theta \text{ WHERE } S) \not\equiv B_1(\theta \text{ WHERE } S) \vee B_2(\theta \text{ WHERE } S)$$

Now we introduce notation which allows us to think of terms and substitutions as of subsets of (a power of) an ω-Herbrand universe.

Definition 18 (Terms and Substitutions as Subsets of $U^\omega_\mathcal{L}$).
Let t be a term of a language $\mathcal{L}$. Let $\theta = \{t_i/x_i \mid i < \omega\}$ be a substitution in $\mathcal{L}$, and let $\tilde{\theta} = \langle t_0, t_1, t_2, \ldots\rangle$. We define:

1. $[\![t]\!] = \{t[\nu] \in U^\omega_\mathcal{L} \mid \nu : Var \longrightarrow U^\omega_\mathcal{L}\}$.
2. $[\![\theta]\!] = \{\tilde{\theta}[\nu] \in (U^\omega_\mathcal{L})^\omega \mid \nu : Var \longrightarrow U^\omega_\mathcal{L}\}$.
3. $[\![\theta \text{ WHERE } S]\!] = \{\tilde{\theta}[\nu] \in (U^\omega_\mathcal{L})^\omega \mid U^\omega_\mathcal{L} \models S[\nu]\}$.

For instance: $[\![f(c_1, g(c_2))]\!] = \{f(c_1, g(c_2))\}$, $[\![g(x)]\!] = \{g(e) \mid e \in U^\omega_\mathcal{L}\}$, $[\![x]\!] = U^\omega_\mathcal{L}$.

Sometimes we may write, say, $[\![\theta_1]\!] \subseteq [\![\theta_2]\!]$ without specifying the language of the substitutions. Such a departure from the formal style does not lead to any problems: if θ_1, θ_2 are substitutions in language $\mathcal{L}$, and if $\mathcal{L} \subseteq \mathcal{L}'$, then the relation $[\![\theta_1]\!] \subseteq [\![\theta_2]\!]$ does not depend on whether the substitutions are interpreted as subsets of $U^\omega_\mathcal{L}$ or of $U^\omega_{\mathcal{L}'}$.

The choice of the ω-Herbrand universe in the definition above is essential. One can see that even in the case of a language with a single individual constant c and no function symbols, $[\![c]\!] \neq [\![x]\!]$. There would not be any difference between $[\![c]\!]$ and $[\![x]\!]$ if one defined these notions using the conventional Herbrand universe.

Example 4.
1. $[\![t_1]\!] \subseteq [\![t_2]\!]$ iff there exists a substitution θ such that $t_1 = t_2\theta$.
2. $[\![t_1]\!] \cap [\![t_2]\!] = [\![t]\!]$ iff t is the result of a most general unification of t_1 and t_2.
3. $[\![\theta \text{ WHERE } \top]\!] = [\![\theta]\!]$.
4. $[\![\theta \text{ WHERE } S]\!] \subseteq [\![\theta]\!]$.
5. If $CET^\omega_\mathcal{L} \vdash S_1 \rightarrow S_2$ then $[\![\theta \text{ WHERE } S_1]\!] \subseteq [\![\theta \text{ WHERE } S_2]\!]$.
6. If $(\theta \text{ WHERE } S)$ is a canonical guarded substitution then $[\![\theta \text{ WHERE } S]\!] \neq \emptyset$.
7. $[\![\theta_1]\!] \subseteq [\![\theta_2]\!]$ iff there exists a substitution θ such that $\theta_1 = \theta_2\theta$
(which means that θ_2 is a more general substitution than θ_1.)

The following theorem points out one more difference between the two versions of Herbrand universes: it holds in the setting of ω-Herbrand universes, it would not be true in the setting of conventional Herbrand universes.

Theorem 19. (Strong Compactness)
1. If $[\![t]\!] \subseteq \bigcup_{i\in I}[\![t_i]\!]$ then there exists $i_0 \in I$ such that $[\![t]\!] \subseteq [\![t_{i_0}]\!]$.
2. If $[\![\theta]\!] \subseteq \bigcup_{i\in I}[\![\theta i]\!]$ then there exists $i_0 \in I$ such that $[\![\theta]\!] \subseteq [\![\theta_{i_0}]\!]$.

The following lemma will allow us to apply what we already know about normal forms in $CET^{\omega}_{\mathcal{L}}$ in order to obtain results about guarded substitutions.

Lemma 20. *Let $\theta = \{t_i/x_i \mid i \in I\}$ be a finite substitution and let S be a guard. Then:*

$$[\![\theta \text{ WHERE } S]\!] = \{\tilde{\nu} \in (U^{\omega}_{\mathcal{L}})^{\omega} \mid U^{\omega}_{\mathcal{L}} \models \exists \boldsymbol{y}(\textstyle\bigwedge_{i\in I} x_i{=}t_i(\boldsymbol{y}) \wedge S(\boldsymbol{y}))[\nu]\}$$

Using this lemma we immediately obtain normal form theorems for guarded substitutions, analogous to those for formulas of CET^{ω}.

Theorem 21 (Normal Form Theorem).
For every finite guarded substitution $(\theta \text{ WHERE } S)$, there exists a finite number of finite canonical guarded substitutions $(\theta_i \text{ WHERE } S_i) : i \leq k$ such that

$$[\![\theta \text{ WHERE } S]\!] = \textstyle\bigcup_{i\leq k}[\![\theta_i \text{ WHERE } S_i]\!].$$

(If $k{=}0$ the right side of this equality is interpreted as $\emptyset$.) The substitutions $(\theta_i \text{ WHERE } S_i) : i \leq k$ can be effectively obtained from $(\theta \text{ WHERE } S)$. Moreover if S is a positive formula then each of the guards S_i in the resulting substitutions is $\top$.

Corollary 22. *For finite guarded substitutions it is decidable whether $[\![\theta \text{ WHERE } S]\!] \neq \emptyset$.*

7 COMPUTING ANSWER SUBSTITUTIONS

This section is a preparatory step towards using the equality theory CET^{ω} to generate computed answer substitutions in a guard logic programming language.

In the following theorems we consider formulas B which contain arbitrary predicate symbols.

Theorem 23 (Computing Substitutions).
For any language $\mathcal{L}$, there exists an algorithm which takes as input a positive formula with guards $B(\boldsymbol{x})$, and returns as output either the statement: "$U^{\omega}_{\mathcal{L}} \not\models \exists \boldsymbol{x} B(\boldsymbol{x})$" if it is correct, or otherwise returns a finite number $k > 0$ of finite canonical guarded substitutions $(\theta_i \text{ WHERE } S_i) : i \leq k$ such that

$$U^{\omega}_{\mathcal{L}} \models B(\boldsymbol{x})(\theta \text{ WHERE } S) \quad \textit{iff} \quad [\![\theta \text{ WHERE } S]\!] \subseteq \textstyle\bigcup_{i\leq k}[\![\theta_i \text{ WHERE } S_i]\!].$$

and after that the algorithm terminates. Moreover if $B(\boldsymbol{x})$ is a positive formula with positive *guards than in the output substitutions each of the guards S_i is $\top$.*

As explained in the Introduction, algorithms from the theorems above can supplement a resolution system, and form a base of a logic programming language that overcomes certain deficiencies of currently existing languages. In our language the symbol $=$ would mean neither unifiability nor syntactical identicalness, it would mean equality, as understood in ω-Herbrand interpretations.

For the proof of the theorems in this section, we replace in B the predicates different from $=$ $\perp$, and then by the Normal Form Theorem 11 we transform the formula to a disjunction of canonical formulas. The canonical formulas determine in a straightforward way the necessary substitutions.

8 CONCLUSION

We developed a system of "true equality" constraints over an open universe of terms (finite trees). Our constraints, called guards, are built from equalities between terms and may contain all Boolean operations and existential and universal quantifiers. The entities satisfying guards are described by means of guarded substitutions. We found an algorithm which given a guard, generates all such substitutions.

Unlike in currently implemented logic programming languages our approach guarantees that equality is understood as in the first order classical logic and interpreted in a fully declarative way.

In further research we will implement a constraint logic programming language based on the ideas of this paper. We will also investigate possibilities of combining our constraint solving mechanism with mechanisms which solve constraints over other domains. Still further work will be devoted to parallelization.

Our results complement those of Kunen, Shepherdson, Mancarella, Martini and Pedreschi which concerned the closed universe of terms.

References

1. K. R. Apt, Introduction to Logic Programming, in J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, North Holland, 1989.
2. K. L. Clark, Negation as Failure, in *Logic and Databases*, H. Gallaire and J. Minker (eds.), Plenum Press, New York, 1978, 193-322.
3. K. L. Clark, Negation as Failure, in: H. Gallaire and J. Minker (eds) *Logic and Databases*, Plenum Press, New York, 1978, pp. 293-322.
4. A. Colmerauer, Prolog and Infinite Trees, in K.L. Clark and S.A. Tarnlund (eds) *Logic Programming*, Academic Press, New York, 1982, pp. 231-251.
5. A. Colmerauer, Solving Equations and Inequations on Finite and Infinite Trees, in *Proceedings of the Conference on Fifth Generation Computer Systems*, Tokyo, 1984, pp. 85-99.
6. A. Colmerauer, An Introduction to Prolog III, *Comm. ACM* 33, 7, July 1990.
7. A. Colmerauer, Theoretical Model of Prolog II, in: M. Van Caneghem and D.H.D. Warren (eds), Albex Publishing Corp, 1986.
8. J. Jaffar, J.-L. Lassez, Guard Logic Programming, in: *Proceedings of the* 14^{th} *ACM Symposium of the Principles of Programming Languages*, Munich, 1987, pp. 111-119.
9. J. Jaffar, J.-L. Lassez and M. J. Maher, A Logic Programming Language Scheme, in: D. DeGroot and G. Lindstrom (eds.), *Logic Programming, Functions, Relations and Equations*, Prentice-Hall, 1986, pp. 441-468.
10. K. Kunen, Some Remarks on the Completed Database, in: R. A. Kowalski and K. A. Bowen (eds.), *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, MIT Press, 1988, pp. 978-992.
11. K. Kunen, Negation in Logic Programming, *Journal of Logic Programming* 1987, No 4, pp 289-308.
12. K. Kunen, Signed Data Dependencies in Logic Programs, *Journal of Logic Programming*, 1989, vol. 7, No 3, pp.231-247.
13. J. W. Lloyd, *Foundations of Logic Programming*, Second extended edition, Springer Verlag, 1987.
14. P. Mancarella, S. Martini, D. Pedreschi, Complete Logic Programs with Domain-Closure Axiom, *Journal of Logic Programming*, 1988, vol. 5, No 3, pp.263-276.

15. J. A. Plaza, *Fully Declarative Programming with Logic – Mathematical Foundations*, (Doctoral Dissertation,) City University of New York, July 1990.
16. J. A. Plaza, Operators on Lattices of ω-Herbrand Interpretations, *Proceedings of the Logical Foundations of Computer Science Symposium "Logic at Tver"*, Lecture Notes in Computer Science, Springer Verlag, 1992,
17. J. A. Plaza, Fully Declarative Logic Programming, *Proceedings of the Conference on Programming Languages Implementation and Logic Programming (PLILP'92)*, Lecture Notes in Computer Science, Springer Verlag, 1992.
18. H. Rasiowa and R. Sikorski, Algebraic treatment of the notion of satisfiability, *Fundamenta Mathematicae 40* (1953), pp. 62-95.
19. J. C. Shepherdson, Negation in Logic Programming, in: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 1988, pp.19-88.
20. J. C. Shepherdson, Language and Equality Theory in Logic Programming, Report No. PM-91-02, University of Bristol.

This article was processed using the LaTeX macro package with LLNCS style