Types in Unification Theory

Obrad Kasum

June 11, 2015

Abstract

When the syntactical unification is considered, for two unifiable terms there is always a unifier which is more general (smaller) than every other (an MGU). However, in a general case of E-unification, there is no MGU. Instead, we work with (minimal) dense sets of unifiers. The type of unification problem is determined by the minimal cardinality of a minimal dense set of unifiers. We present here the theory of types in a way which differs from usual.

1 Introduction

The idea of unification appeared in Herbrand's works, but the (syntactical) unification is formally presented in [Robinson 1965] for the first time. It is the basis of the resolution principle, the main principle in automated theorem proving. The E-unification is a natural generalization of the syntactical unification.

In section 2, basic facts about syntactical unification are presented along with the first, most intuitive, algorithm for it. In section 3, E-unification and types of unification problems and equational theories are introduced and some examples are given. The theorem 3.1 is a result of this paper.

Papers [Baader 1989], [Büttner 1988], [Franzen 1992] consider types in some special cases.

2 Syntactic Unification

Term "syntactic unification" refers to the process of making two terms syntactically identical by substituting their variables with terms. In this subsection, we will formally present the problem of the syntactical unification and give the basic algorithm which solves it.

Let $\mathcal{L}$ be a first-order language[1] without relational symbols, $\mathcal{V}$ be the set of all variables and $\mathcal{T}$ be the set of all terms of $\mathcal{L}$. A *substitution* is a function from $\mathcal{V}$ to $\mathcal{T}$ which is equal to the identity mapping almost everywhere. For a substitution σ, we define its extension $\overline{\sigma}$ from $\mathcal{V}$ to $\mathcal{T}$ by induction on length of terms as follows:

1. $\overline{\sigma}(x) = \sigma(x)$, for a variable x;
2. $\overline{\sigma}(c) = c$, for a constant c;
3. $\overline{\sigma}(t) = f(\overline{\sigma}(t_1), \ldots, \overline{\sigma}(t_n))$, for a term $t = f(t_1, \ldots, t_n)$.

[1] In practice, this language is countable.

We will denote $\overline{\sigma}$ also with σ and write $t\sigma$ for $\sigma(t)$. According to this convention, the composition of substitutions σ and θ in t should be denoted by $t\sigma\theta$, so we will also introduce the convention of denoting the composition of those substitutions by $\sigma\theta$ (instead of standard $\theta\sigma$). If $\{x_1, \ldots, x_n\}$ is the set on which substitution σ differs from identity mapping, than we shall write this substitution as $\{x_1 \mapsto x_1\sigma, \ldots, x_n \mapsto x_n\sigma\}$. Two terms t and s are said to be unifiable if there is a substitution σ such that $t\sigma = s\sigma$. An *unification problem* is a pair of terms (for which we are trying to find out if they are unifiable). When we consider a syntactical unification problem, the default language in which we work is the minimal one, unless it is said otherwise.

Example 2.1. Terms $t = f(t_1, \ldots, t_n)$ and $s = g(s_1, \ldots, s_m)$, where t_i and s_i are some terms and f and g are different functional symbols, are not unifiable. For, every $t\sigma$ begins with symbol f, while $s\sigma$ begins with g, so they cannot be equal.

If t is a proper subterm of s, than every $t\sigma$ is a proper subterm of $s\sigma$ and we have again that t and s are ununifiable.

Example 2.2. Terms $f(g(x), g(h(z)))$ and $f(z, g(y))$ are unifiable. For,

$$\sigma = \{y \mapsto h(g(x)), z \mapsto g(x)\}$$

is unifier for them. It can be easily seen that some θ is an unifier for these terms iff $\theta = \sigma\rho$, for some ρ.

It is obvious that if two terms are unifiable, they have infinitely many unifiers. However, as in example 2.2, there is a unifier from which all others may be obtained. We will call it a *most general unifier* or an *MGU*.

Definition 2.1. *A substitution σ is more general than a substitution θ (we write $\sigma \preccurlyeq \theta$) if there is a substitution ρ such $\theta = \sigma\rho$. An MGU for terms t and s is a unifier for them which is more general than every other.*

The relation $\preccurlyeq$ is quasi-ordering of the set of all substitutions (it is reflexive and transitive) and $\approx = \preccurlyeq \cap \succcurlyeq$ is equivalence relation on that set. There is a natural characterisation of $\approx$ which relies on a notion of a variable-renaming substitution. Those substitutions are exactly all bijective substitutions from $\mathcal{V}$ to $\mathcal{V}$. Now we have that $\sigma \approx \theta$ iff there is a variable-renaming substitution ρ such that $\sigma = \theta\rho$. The proof of this result can be found in [Lassez, Maher, Mariott 1987]. We are ready to state the main theorem about MGUs.

Theorem 2.1. *For unifiable terms t and s, there exists MGU and it is unique up to $\approx$.*

For practical purposes, it is of great importance to know an algorithm for checking if two terms are unifiable and finding an MGU if they are. We present here a pseudo-code of the most intuitive one (firstly described in [Robinson 1965]), although it is not very efficient. For more efficient algorithms, a proof of correctness of the algorithm given here and its analysis see [Baader, Snyder 1999]. An implementation of an unification algorithm is given in [Prešić 1991].

The algorithm uses an algorithm for finding the composition of two substitutions. One of these is obvious from the next description of the composition. For two terms σ and θ let $\mathcal{V}_\sigma = \{v \in \mathcal{V} : v\sigma \neq v\}$ and $\mathcal{V}_\theta = \{v \in \mathcal{V} : v\theta \neq v\}$. Now we have

$$\sigma\theta = \{\langle v, v\sigma\theta\rangle : v \in \mathcal{V}_\sigma\} \cup \{\langle v, v\theta\rangle : v \in \mathcal{V}_\theta \setminus \mathcal{V}_\sigma\} \cup \mathrm{id}_{\mathcal{V}\setminus(\mathcal{V}_\sigma \cup \mathcal{V}_\theta)}.$$

The unification algorithm mentioned above is following. Be aware of that the function is closed after "return" (like in "C") and that the function returns '/' when t and s are not unifiable.

```
substitution mgu(term t, term s)
    begin
    if t = s = x then return {};
    if t = x and x occurs in s then return '/';
    if t = x then return {x ↦ s};
    if s = x and x occurs in t then return '/';
    if s = x then return {x ↦ t};
    if t = f(t_1, ..., t_n) and s = g(s_1, ..., s_m) then
        if f ≠ g then return '/';
        else
            begin
            substitution σ := {};
            for i := 1 to n do
                begin
                t_i := t_iσ;
                s_i := s_iσ;
                θ := mgu(t_i,s_i);
                if θ = '/' then return '/';
                σ := σθ;
                end
            return σ;
            end
    end
```

3 E-unification

When E-unification is considered, the intention is to make two terms equal *modulo* E by applying a substitution. This is a generalization of syntactical unification. All symbols introduced in previous subsection have same meaning here.

Let E be a theory[2] over a language $\mathcal{L}$ consisting only of identities. These theories are called *equational*. Two terms s and t are said to be *equal modulo* E (denoted by $s =_E t$) if $E \vdash s = t$ and they are said to be unifiable modulo E if there exists a substitution σ such that $s\sigma =_E t\sigma$. An E-unification problem is (again) a pair of terms. It is clear that if $E = 0$, we are talking about syntactical unification.

The following quasi-ordering is of interest here.

Definition 3.1. *A substitution σ is more general modulo E than a substitution θ (denoted by $\sigma \preccurlyeq_E \theta$) if there is a substitution ρ such that $x\theta =_E x\sigma\rho$ for all $x \in \mathcal{V}$. The corresponding equivalence relation is denoted by $\approx_E$.*

Example 3.1. Let C_f be the theory consisting of the axiom $f(x, y) = f(y, x)$ (commutativity) over the language $\mathcal{L} = \{f, a, b\}$ (a and b are constant symbols). Let us consider terms $s = f(x, y)$ and $t = f(a, b)$. Obviously, they are unifiable modulo C_f and $\sigma = \{x \mapsto a, y \mapsto b\}$ and $\theta = \{x \mapsto b, y \mapsto a\}$ are unifiers for them. Every other unifier is less general than one of them. Since they are incomparable, these terms have no MGU.

When considering E-unification, instead of an MGU, we are interested in a set which has a similar property of *generality* - each unifier for the terms should be less general than

[2]In practice, E is a recursive theory.

a unifier from the considered set. These sets are naturally called *dense*. Among them, particularly interesting are *minimal* ones - those containing no comparable elements (antichains).

Example 3.2. If terms s and t are syntactically unifiable and if σ is an MGU, than $\{\sigma\}$ is a minimal dense. This shows that the minimal dense sets generalize MGUs.

In example 3.1 $\{\sigma, \theta\}$ are minimal dense.

Example 3.3. Let AI_f be a theory consisting of axioms $f(x, f(y, z)) = f(f(x, y), z)$ (associativity) and $f(x, x) = x$ ("f is idempotent") over the language $\mathcal{L} = \{f\}$. In [Baader 1986] it is shown that unification problem with terms $f(x, f(y, x))$ and $f(x, f(z, x))$ has no minimal dense set of unifiers.

For this reason we introduce an object ∞ greater than all cardinals which allows us to make the following definition.

Definition 3.2. *The type of an E-unification problem (s, t) is defined with*

$$\tau_{s,t} = \min\{|D| : D \text{ is a minimal dense set of unifiers for } (s, t)\}.$$

The type of a theory E is defined with

$$\tau_E = \sup\{\tau_{s,t} : (s, t) \text{ is an } E\text{-unification problem}\}.$$

In practice, we consider only countable languages, so a type is an element of the set $[0, \omega] \cup \{\infty\}$.

Example 3.4.

1. Terms s and t are E-unifiable iff $\tau_{s,t} > 0$.
2. The type of a syntactical unification problem is 1 if the terms are unifiable and 0 if they are not (see example 3.2). According to this, the type of syntactical unification is $\tau_0 = 1$.
3. The type of the theory AI_f is ∞ (see example 3.3).
4. The unification problem in example 3.1 is of type 2.

In preceding, a constant symbol is considered to be 0-nary functional symbol.

Definition 3.3. *A term-pattern for language $\mathcal{L}$ is the mapping from the set of all functional symbols to ω that differs from 0 only in finitely many points. A pattern of term t, denoted by F_t, is a term-pattern which assigns to each functional symbol the number of times that it occurs in t. The variable-number of a term-pattern F is defined to be*

$$vF = \max\{\text{the number of variables in } t : F_t = F\}.$$

Equational theory E is said to be pattern-preservative if for every identity $t = s$ in it, the equation $F_t = F_s$ is satisfied.

Theorem 3.1. *If E is a pattern-preservative equational theory, than $\tau_E < \infty$.*

Proof. Let s and t be E-unifiable terms, U be the set of all unifiers for them and M be a set of all minimal unifiers for them, with no equivalent unifiers in it. Let σ be a unifier for these terms. Assume that $\sigma = \{x \mapsto t\}$ (the following is easily adjustable to the general case). Let $v_1, \ldots, v_{vF_t}$ be different variables, different from x. Consider the set N of all unifiers of the form $\{x \mapsto s\}$ which are more general than σ, where all variables of s are among variables v_i. Obviously $F_s \leqslant F_t$, so N is finite and non-empty and, consequently, it contains a $\preccurlyeq_E$-minimal element θ. θ is equivalent to an element of M. Otherwise, there is $\theta_1 \prec_E \theta$, so $\theta_2 = \{x \mapsto x\theta_1\} \prec_E \theta$ is equivalent to an element of N. This contradicts to the minimality of θ. We have just shown that M is dense. It is obvious that it is minimal, so the proof is complete.

□

Example 3.5.

1. By previous theorem, theory C_f from example 3.1 is of type $\leqslant \omega$. We will show that its type is exactly ω. For, consider the unification problem of terms s and t determined by full binary trees S and T of hight $n+2$, whose all elements except leaves are equal to f, leaves of first are $x_1, y_1, \ldots, x_{2^n}, y_{2^n}$ and leaves of second are $a, b, \ldots, a, b$. The set D of all substitutions of the form $\{x_1 \mapsto \alpha_1, y_1 \mapsto \beta_1, \ldots, x_{2^n} \mapsto \alpha_{2^n}, y_{2^n} \mapsto \beta_{2^n}\}$, where $\{\alpha_i, \beta_i\} = \{a, b\}$, is minimal dense and every other minimal dense set is obviously of same cardinality as D. So, $\tau_{s,t} = |D| = 2^{2^n}$ and, consequently, $\tau_{C_f} = \omega$.

2. Let us consider the theory A consisting of the axiom $f(f(x, y), z) = f(x, f(y, z))$ over the language $\mathcal{L} = \{f, a\}$ and the unification problem of terms $s = f(x, a)$ and $t = f(a, x)$. It is obvious that each dense set of unifiers must contain an element σ such that $x\sigma =_A f(a, f(a, \ldots f(a, a) \ldots))$, where the term on the right side is of any possible length. Hence, a dense set must be infinite, so $\tau_{s,t} \geqslant \omega$. By previous theorem, $\tau_A < \infty$ and we have $\tau_{s,t} = \tau_A = \omega$.

4 Conclusion and Discussion

For equational theory E, the information about its type should give us the first insight in the unification in that theory. A better insight should be obtained by classifying sets of problems by their types. We pointed out to some central identities which should be considered and gave some facts about theories containing them. Theories of semigroups (mentioned here), monoids, Boolean algebras are among those of interest in this subject.

References

[Baader 1986] F. Baader: Unification in idempotent semigroups is of type zero. Journal of Automated Reasoning 2 (3), 1986, 283-286.

[Baader 1989] F. Baader: Characterisations of unification type zero. Proceedings of 3rd International Conference on Rewriting Techniques and Applications (N. Dershowitz, ed.), Vol. 355 of Lecture Notes in Computer Science, Springer-Verlag, Chapel Hill, North Carolina, 1989, 2-14.

[Baader, Snyder 1999] F. Baader, W. Snyder: Unification theory. Handbook of automated reasoning (A. Robinson, A. Voronkov, eds.), Elsevier Science Publishers B. V., 1999, 445-533.

[Büttner 1988] W. Büttner: Unification in finite algebras is unitary (?). Proceedings of 9th International Conference on Automated Deduction (E. Lusk, R. Overbeek, eds.), Vol. 310 of Lecture Notes in Computer Science, Springer-Verlag, Argonne, IL, 1988, 368-377.

[Franzen 1992] M. Franzen: Hilbert's tenth problem is of unification type zero. Journal of Automated Reasoning 9, 1992, 169-178.

[Lassez, Maher, Mariott 1987] J.-L. Lassez, M. Maher, K. Mariott: Unification revised. Foundations of Deductive Databases and Logic Programming (J. Minker, ed.), Morgan Kaufman, Los Altos, California, 1987, 587-625.

[Prešić 1991] S. B. Prešić: Algoritam ujednačavanja. Računarstvo, Vol. 1, No. 1, Matematički institut, Belgrade, 1991, 35-46.

[Robinson 1965] J. Robinson: A machine oriented logic based on the resolution principle. Journal of ACM 12 (1), 1965, 23-41.