Some notes on unification (fragment)

Dan Dougherty
Wesleyan University

1 Terms and substitutions

Definition 1 A *signature* is a countable set Σ of *function symbols*, each with a specified *arity*, which indicates the number of arguments the function takes.

Definition 2 Fix a signature Σ and fix a countably infinite set X of *variables*, disjoint from Σ. The set $T_\Sigma(X)$ of *Terms* (determined by Σ and X) is the least set satisfying:

- if $x \in X$ then $x \in T_\Sigma(X)$.
- if f has arity n and $\forall i \leq n\ t_i \in T_\Sigma(X)$ then $f(t_1, \ldots, t_n) \in T_\Sigma(X)$.

Here $f(t_1, \ldots, t_n)$ is, abstractly, the ordered pair consisting of the symbol f and the list of arguments $(t_1, \ldots t_n)$. There are various ways of concretely representing terms, some good for parsing by humans, some good for processing by machines. We will not concern ourselves with this issue (except insofar as we just chose one method, for writing terms on the page!). We use the symbol $\equiv$ to express syntactic identity (as opposed to, for example, equality relative to a given theory).

It is customary when a symbol a has arity 0 to refer to it as a *constant symbol*, and to write simply a instead of $a()$.

Unless indicated otherwise, variables will be denoted $x, y, z, \ldots$, constants by $a, b, c, \ldots$.

Definition 3 *Vars*(T) is the variables occurring in a term:

- *Vars*$(x) \equiv \{x\}$,
- *Vars*$(f(t_1, \ldots, t_n)) \equiv \bigcup\{\, \mathit{Vars}(t_i) \mid 1 \leq i \leq n\}$.

The following notion will prove to be a great technical convenience: A substitution θ is *idempotent* if $\theta\theta \equiv \theta$ (i.e., for all x, $\theta(\theta x) \equiv \theta x$).

It is easy to recognize idempotent substitutions:

Lemma 4 *A substitution θ is idempotent iff $D\theta \cap I\theta = \emptyset$.* ///

One of the nice properties of idempotent substitutions is the following.

Lemma 5 *If σ is idempotent then for all θ, $\sigma \preceq \theta$ iff $\theta\sigma \equiv \theta$.* ///

We may usually work with idempotent substitutions with no loss of generality, since every substitution is equivalent under renaming to an idempotent one:

Lemma 6 *For all σ there exists σ' such that $\sigma' \simeq \sigma$ and σ' is idempotent.* ///

Syntactic Unification

A substitution θ is a (syntactic) *unifier* of two terms s and t if $\theta s \equiv \theta t$ (we say that s and t are (syntactically) *unifiable*). Often the qualifier "syntactic" is suppressed.

Example 7

- The pair $[s, s]$ is unified by any substitution (in particular, by the identity).
- If $x \notin \mathit{Vars}(t)$ then the pair $[x, t]$ is unified by the substitution $x \mapsto t$.
- If $x \in \mathit{Vars}(t)$ then the pair $[x, t]$ has no unifiers.
- The pair $[f(s_1, \ldots, s_n), g(t_1, \ldots, t_k)]$ has no unifiers.
- The pair $[f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)]$ is unified by θ iff θ unifies each of the pairs $[s_1, t_1], \cdots [s_n, t_n]$.

The example involving $[f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)]$ suggests that we should focus attention on unifying several pairs simultaneously. To this end we make the following definition.

Definition 8 A *pair* is a two-element multiset of terms. A *system* is a finite set $\mathcal{S}$ of pairs. A *unifier* of a system is a simultaneous unifier of the pairs in the system.

It is convenient to write $\mathcal{S}, [s,t]$ instead of $\mathcal{S} \cup \{[s,t]\}$. Since this is ambiguous as a decomposition of the system in question ($\mathcal{S}$ may or may not contain $[s,t]$), we introduce the notation $\mathcal{S}; [s,t]$ to refer to $\mathcal{S} \cup \{[s,t]\}$ with the understanding that $[s,t]$ is *not* a pair in $\mathcal{S}$.

Note that if σ unifies $[s,t]$ and $\sigma \preceq \theta$, then θ unifies $[s,t]$. This applies to systems as well, so we can say that the set of unifiers of a system is closed upwards in the $\preceq\cdot$ ordering on substitutions. What more can we say in general about a set of unifiers?

We saw that a pair $[x,B]$ with $x \notin \mathit{Vars}(B)$ is unified by the substitution sending x to B — let us call this substitution σ. Note that σ is idempotent. Furthermore we will see that for *any* unifier θ of $[x,B]$, $\sigma \preceq \theta$. In fact it will turn out that any system which is unifiable will possess unifiers such as σ, i.e., unifiers which are minimal in the $\preceq$ ordering. In light of the observation above that the set of unifiers of a system is closed upwards (with respect to $\preceq\cdot$), this implies that the set of unifers of a unifiable system can be completely characterized once we have computed a most general one.

Definition 9 A substitution σ is a *most general unifier* of system $\mathcal{S}$ if

- θ is an idempotent unifier of $\mathcal{S}$,
- $D\sigma \subseteq \mathit{Vars}(\mathcal{S})$, and
- for every unifier θ of $\mathcal{S}$, $\sigma \preceq \theta$.

We have passed to systems because we have seen that when trying to unify a single pair we are led to *simultaneously* unifying several pairs. It would be very pleasant if we could reduce this simultaneous problem to an incremental one. Now, given a system $\mathcal{S}$ of the form

$$[s_1,t_1],\ [s_2,t_2], \cdots$$

we cannot simply unify the first pair, then the second, etc., because there is no reason to think that the individual answers will be consistent with each other. But suppose we were able to compute a *most general unifier* of $[s_1,t_1]$, say σ. Now let θ be any unifier of the entire system. In particular, θ unifies $[s_1,t_1]$, so $\sigma \preceq \theta$. Since σ is idempotent, this implies that $\theta\sigma \equiv \theta$. But this in turn implies that, not only does θ unify the original system $\mathcal{S}$, but θ unifies $\sigma\mathcal{S}$. In other words, we can *apply* the most general unifier

of a part of a system to the entire system, without affecting the space of solutions. We can then proceed to unify $[\sigma s_2, \sigma t_2]$ and so on. This approach will be generalized and presented formally below.

Another way to describe this approach: construct a solution to the entire problem by building upon (composing!) solutions to individual parts. This is a common problem-solving strategy, but the key point here is that the existence of most general solutions to the individual sub-problems means that *we never have to backtrack over individual solutions*, since they will never be inconsistent with each other. (Compare this with a classic backtracking problem such as the eight queens.)

This is a fundamental property of most general unifiers, especially when unification is part of a more complex process (such as theorem-proving).

Term-pairs such as $[x, t]$ represent very easy unification problems, since we can characterize the set of solutions immediately, as the set of generalizations of $\sigma \equiv x \mapsto t$. Taking our cue from the above discussion, if we find a pair $[x, t]$ in a system we can apply σ to the rest of the system and consider the pair $[x, t]$ to be "solved" (and know that we have found part of our solution).

Term-pairs in which neither element is a variable are either immediately seen to be non-unifiable or are to be decomposed into sub-problems. Clearly we may iterate this process until we discover non-unifiability or are left with pairs with a variable-element.

This suggests an algorithm:

1. Recursively decompose pairs until a pair with different head-symbols appears (halt with failure) or until each pair is of the form $[x, t]$ (go to 2).

2. For each "unsolved" pair $[x, t]$, apply the substitution $x \mapsto t$ to the rest of the system and mark the pair $[x, t]$ as "solved" (go to 1).

We would like to know that this algorithm halts and that the partial answers given by the $x \mapsto t$ combine to give the answer we want. In the next subsection we do even better than that.

Exercise 10 Show that the algorithm above always halts. [Hint: show that each step in the loop halts, and that each loop iteration decreases the number of "unsolved" variables.]

Transformations for unification

Martelli and Montanari [MM82] defined a set of *transformations* in order to study syntactic unification. Transformation-based unification methods attempt to reduce systems representing unification problems to *solved* systems, from which solutions may be extracted immediately. One way to think of transformations is that they abstract the computational core of an algorithm from its control structure.

A pair $[x, t]$ is *solved* in $\mathcal{S}$, and x is a *solved variable* of $\mathcal{S}$, if there are no occurrences of x anywhere except for the left-hand x in $[x, t]$. If each pair in $\mathcal{S}$ is solved then $\mathcal{S}$ is a *solved system* and determines an idempotent substitution in an obvious way, although a pair consisting of two distinct solved variables requires a choice as to which of them is to be in the domain of the substitution. We will assume that a uniform method exists for making such a choice, and so will refer to *the* substitution determined by a solved system.

If σ is an idempotent substitution, write $[\sigma]$ for any solved system by which it is determined. The key fact about solved systems is that they embody their own most general unifiers; specifically, we will prove below that if $[\sigma]$ is solved then σ is a most general unifier of $[\sigma]$.

Thus we can find unifiers of a system $\mathcal{S}$ if we can find a *solved* system whose set of unifiers is the same as that of $\mathcal{S}$. The strategy, then, is to iteratively apply transformations to $\mathcal{S}$ in such a way that the set of unifying substitutions is preserved, and hope we arrive at a solved sytem eventually. An analogy is provided by the Euclidean Algorithm for computing greatest common divisors.

The following variant of Martelli and Montanari's transformations is defined by Gallier and Snyder in [GS89]

Definition 11 The following comprise the set of transformations for *Syntactic Unification.*

- *Trivial:*

$$\mathcal{S};\ [s, s] \Longrightarrow \mathcal{S}$$

- *Term Decomposition:*

$$\mathcal{S};\ [f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)] \Longrightarrow \mathcal{S},\ [s_1, t_1], ..., [s_n, t_n]$$

- *Variable Elimination:*

$$\mathcal{S};\ [x,t] \Longrightarrow \varphi\mathcal{S},\ [x,t], \quad \text{where } \varphi \text{ is the substitution } \{x \mapsto t\},$$

  provided $[x,t]$ is not solved and x does not occur in t.

Observe the use of ";" on the left-hand sides of transformations, so that the effect of the transformation is unambiguous, and the use of "," on the right-hand sides, to preclude repetition of identical pairs.

These transformations naturally define a non-deterministic procedure, which we denote *Syntactic Unification*, or simply SU. Write $\mathcal{S} \stackrel{SU}{\Longrightarrow} \mathcal{S}'$ if $\mathcal{S}'$ is obtained from $\mathcal{S}$ in one SU step.

Theorem 12 (Martelli and Montanari 1982) *Every* SU *computation terminates. If $\mathcal{S}$ is unifiable then every* SU *computation on $\mathcal{S}$ terminates in a solved system determining a most general unifier for $\mathcal{S}$. If $\mathcal{S}$ is not unifiable then no* SU *computation on $\mathcal{S}$ terminates in a solved system.*

The theorem follows from the following lemmas.

Lemma 13 *Every sequence of* SU *transformations terminates.*

Proof. Associate with each system the number of unsolved variable occurrences and then the sum of the depths of the terms; order these pairs lexicographically and observe that any transformation decreases the associated pair. ///

Lemma 14 *Suppose $x \notin Vars(t)$. Then the substitution given by $x \mapsto t$ is a most general unifier of the pair $\langle x,t\rangle$.* ///

Lemma 15 *Suppose $\mathcal{S} \stackrel{SU}{\Longrightarrow} \mathcal{S}'$. Then for any substitution θ, θ unifies $\mathcal{S}$ iff θ unifies $\mathcal{S}'$.*

Proof. An easy examination of cases; use the previous lemma for the Variable Elimination case. ///

Lemma 16 *Suppose system $\mathcal{S}$ is irreducible with respect to* SU*. Then $\mathcal{S}$ is unifiable iff it is solved. Furthermore, if $\mathcal{S}$ is solved, with $\mathcal{S} \equiv [\sigma]$ then σ is a most general unifier of $\mathcal{S}$.*

Proof. Easy (use Lemma 14 for the last assertion). ///

It is interesting to note that the the completeness of the above *method* yields information about the nature of the solutions:

Corollary 17 *A unifiable system possesses a most general unifier.*

Note that the algorithm sketched in the previous subsection embodies one particular control structure applied to the Martelli-Montanari transformations.

Corollary 18 *The algorithm in the previous subsection computes most general unifiers.*

Later we will look at "semantic" unification, unification modulo an equational theory.

[Say something about matching, as one-sided unification; use that terminology below in defn of rewriting (aand extended rewriting...]

References

[GS89] Gallier, J. H., and Snyder, W.. Complete sets of transformations for general E-unification. *Theoretical Computer Science* **67**, 1989. 203–260.

[MM82] Martelli, A. and Montanari, U. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems* **4**, 1992. 258–282.