On Unification of Lambda Terms

Giulio Manzonetto[1]
Andrew Polonsky[2]

[1] Université Paris 13, Laboratoire LIPN, CNRS UMR 7030, France
[2] Université Paris Diderot, Laboratoire IRIF, CNRS UMR 8243, France

We propose a unification algorithm for the type-free lambda calculus. As an intermediate step in this development, we are led to the following question:

What does it mean to unify untyped higher-order terms?

As an illustrative example, consider the obsevation, due to Visser, that Ω is the only λ-term which reduces to itself in a single step (up to a redex-free context around it). Wlog, we may assume that the redex is at the root; then any term M with this property must be a solution to a "unification equation"

$$M = (\lambda v.X)Y = X[Y/v] \tag{1}$$

where X,Y are meta-variables for terms.

(Here and henceforth, "=" denotes syntactic α-equality.)

Note that $X = X(v)$ is a higher-order meta-variable — but it is not a "context", because it can have more than one hole.

We may pursue to solve (1) analogously to the usual unification process. Since the substitution result $X[Y/v]$ is an application, we have two possibilities for $X = X(v)$:

$$X(v) = \begin{cases} v & (A) \\ X_1(v)X_2(v) & (B) \end{cases}$$

Case (A), together with (1), yields $(\lambda v.X)Y = X(Y) = Y$, contradicting finiteness of M. (Although, the infinite term $Y = (\lambda v.v)Y$ *is* a solution of (1), and it is indeed a 1-cycle in the infinitary lambda calculus.)

So (A) is ruled out. Rewriting (1) using (B), we obtain

$$(\lambda v.X_1(v)X_2(v))Y = (\lambda v.X)Y = X[Y/v] = X_1(Y)X_2(Y)$$
$$X_1(Y) = \lambda v.X_1(v)X_2(v) \tag{2}$$
$$X_2(Y) = Y \tag{3}$$

Next, we can "destruct" X_1 according to (2), obtaining two possibilities ($X_1(v) = v$ and $X_1(v) = \lambda v'.X_{10}(v,v')$), eliminate one, rewrite the other back into (2), and so forth. Proceeding in this manner, we shall eventually find that the only solution is $M = (\lambda x.xx)(\lambda x.xx)$. (See [2] for other examples.)

In a similar manner, we can enumerate all terms with a specified "subterm pattern", given by a set of unification constraints. Such an algorithm could be used to enumerate all terms with a particular redex structure (for example, in order to analyze possible standardard reductions of shape $Nx \twoheadrightarrow x(Nx)$).

It may not be obvious that the above process should always terminate — unlike the case of first-order unification, substituting a meta-variable does not completely eliminate it. Each iteration develops the root symbol only, and may yet introduce new meta-variables, occurring deeper and more often (while substitution proliferates and rearranges existing ones).

Nevertheless, we prove that the above algorithm does terminate, and yields a set of most general solutions. (There are in general many of them.)

At first, our claim appears to contradict the common wisdom that "higher-order unification is undecidable" [1]. Doesn't it?

First of all, notice that we are working in an untyped setting, whereas higher-order unification is usually considered for simple types. While this does not seem to help much, it may perhaps explain the second difference, which is that *we do not contract the beta redexes.*

Indeed, from the perspective of equational logic, unification is a purely syntactic operation, and is not expected to take into consideration the whole set of axioms a given theory might have.

The lambda calculus is a theory of terms built up with a binary first-order operation and a unary higher-order operation; the beta-rule is merely an axiom of this theory. Generalizing to other higher-order signatures, with perhaps different rewrite rules, one could scarcely hope the generic "unification algorithm" to subsume all their possible equations.

So we do not solve full second-order unification. At the same time, the higher-order meta variables are not inert, either. In a sense, the substitutions being performed at every step are morally contractions of beta-redexes introduced by the algorithm implicitly.

Finally, this approach does not seem to fall under context unification either, since the occurrence of "holes" can be multiple and unbounded.

In the original spirit of the Types meetings, we hope that our presentation may elicit suggestions from the audience toward how our result relates to the well-known unification paradigms.

References

[1] G. Dowek, *Higher-Order Unification and Matching*, Handbook of Automated Reasoning, pp. 1009–1062, Elsevier and MIT Press, 2001

[2] M. Venturini Zilli, *Reduction Graphs in the Lambda Calculus*, Theor. Comput. Sci., Vol. 29, pp. 251–275, 1984.