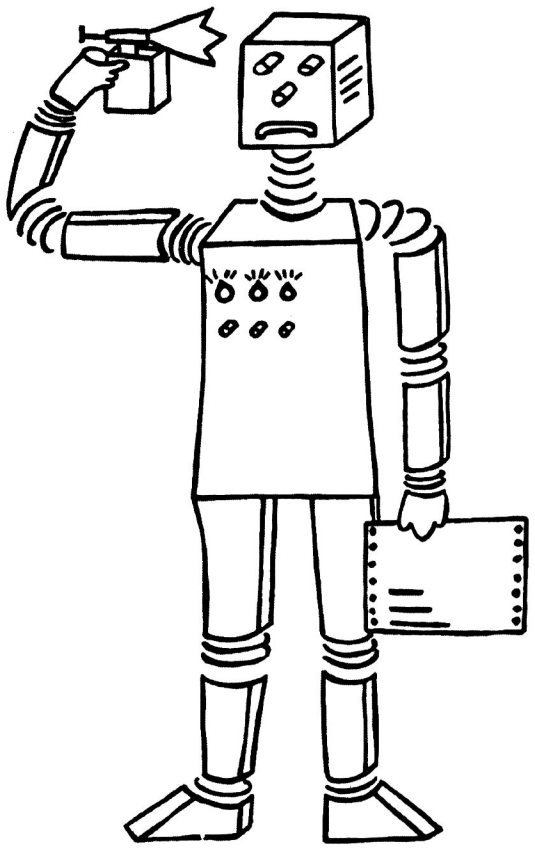
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany

SEKI-Working Paper



Lazy E-Unification - A Method to Delay Alternative Solutions

Hans-Jürgen Bürckert

SEKI-Working Paper SWP-87-07 Okt. 87

Lazy E-Unification - A Method to Delay Alternative Solutions

Hans-Jürgen Bürckert

SEKI-Working-Paper SWP-87-o7 Okt. 87

Lazy E-Unification - A Method to Delay Alternative Solutions*

Hans-Jürgen Bürckert, FB Informatik, Universität Kaiserslautern,

Postfach 3049, D-6750 Kaiserslautern, W.-Germany

net- address: UUCP ...!mcvax!unido!uklirb!buerkert

* extended abstract of a talk given at the 1st Workshop on Unification at Val d'Ajol, France, 1987

One of the most unsuitable properties of E-unification is the existence of more than one most general E-unifiers (G. Plotkin 1972, F. Fages & G. Huet 1986, J. Siekmann 1987). We want to present a general method to delay these alternative solutions in the context of Logic Programming, since there it will be most problematic. Nevertheless, this method is also useful in other kinds of deduction systems (H.J. Ohlbach 1986). The idea is to be lazy in unification, that is to unify at most those parts of a unification problem that will not split up the solution space (H.-J. Bürckert 1986). The partial unifier will be used for resolution and the remaining part of the unification problem will be kept in memory. If the empty clause is derived, the collected residues of the unification problems will be totally E-unified. If they are not E-unifiable, backtracking takes place.

Let Σ be a *signature* consisting of a set *PRED* of *predicate* symbols with their arities, a set *FUN* of *function* symbols with their arities, and an infinite set *VAR* of *variables*. Let $\mathcal{T}$ be the set of *terms* over Σ, let $\mathcal{L}$ be the set of *literals* over Σ and let *SUB* be the set of *substitutions* $\{x_i \leftarrow t_i : 1 \leq i \leq n\}$ over Σ. A *definite clause* is a pair $h \Leftarrow B$ of a literal $h \in \mathcal{L}$ (the *head*) and a finite set B of literals (the *body*) with the obvious meaning that the conjunction of the body literals imply the head literal (all variables are assumed to be universal quantified). A definite clause with empty body is called a *fact*, the others are called *rules*. A *goal clause* is a rule with no head. We use the following notation for these three kinds of *Horn clauses* ($b_1,\ldots,b_n$ is the body B):

- $h \Leftarrow b_1,\ldots,b_n$ (rule)
- h (fact)
- $\Leftarrow b_1,\ldots,b_n$ (goal)

Now, a *logic program* P over Σ is a finite set of rules and facts, and a *query* Q to the program P is any goal clause. We denote the set of variables of any of these or other objects O consisting of terms by $\mathcal{V}(O)$.

We assume a distinguished binary predicate = in *PRED*, written infix and called *equality*. Notice, that both head and body literals might be equality literals. Clauses with equality head are called *functional* clauses, the others are called *relational*. If = should *semantically* denote equality in a program P, then P must contain the *equality* clauses

- $x = x$ (reflexivity)
- $x = y \Leftarrow y = x$ (symmetry)
- $x = z \Leftarrow x = y, y = z$ (transitivity)
- $f(x_1,\ldots,x_n) = f(y_1,\ldots,y_n) \Leftarrow x_1 = y_1,\ldots,x_n = y_n$ (function replaceability)
  for each function symbol $f \in FUN$ with arity $n \geq 1$
- $p(x_1,\ldots,x_n) \Leftarrow p(y_1,\ldots,y_n), x_1 = y_1,\ldots,x_n = y_n$ (predicate replaceability)
  for each predicate symbol $p \in PRED$ with arity $n \geq 1$

Given a goal clause with a *focused* goal literal g and the residue G we call every application of one of the following transformation rules to the goal $g \,\&\, G$ a *goal reduction* with the program P:

(R) $p(t_1,\ldots,t_n) \,\&\, G \longrightarrow s_1 = t_1 \,\&\, \ldots \,\&\, s_n = t_n \,\&\, B \,\&\, G$ *(resolution)*
if $p(s_1,\ldots,s_n) \Leftarrow B$ is a clause of P totally renamed with new variables

(T) $x = x \,\&\, G \longrightarrow G$ *(tautology)*

(B) $x = t \,\&\, G \longrightarrow x = t \,\&\, \{x \leftarrow t\}G$ *(binding)*
if $x \notin \mathcal{V}(t)$, but $x \in \mathcal{V}(G)$ and $\{x \leftarrow t\}$ is a substitution of x by t

(D) $f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n) \,\&\, G \longrightarrow s_1 = t_1 \,\&\, \ldots \,\&\, s_n = t_n \,\&\, G$ *(decomposition)*

(O) $t = x \,\&\, G \longrightarrow x = t \,\&\, G$ *(orientation)*

Notice, that the focused goal literal in the resolution rule might also be an equality literal. A *refutation* of a query Q with a program P is a sequence of goal reductions starting with Q and terminating with a *solved* goal, that is a goal of equality literals $x_1 = t_1,\ldots, x_n = t_n$, such that the t_i are terms and the x_i are pairwise different variables with $x_i \notin \mathcal{V}(t_1,\ldots,t_n)$. Every such solved goal defines a *correct answer* substitution $\{x_1 \leftarrow t_1,\ldots,x_n \leftarrow t_n\}$ to the query Q.

Any ordering of goal reductions is called a *strategy*. If we apply after each resolution step the *unification* rules (T) - (O) to the introduced equality literals $s_1 = t_1 \,\&\, \ldots \,\&\, s_n = t_n$ *(unification conditions)* until they are reduced completely into solved form, then we call this a *standard refutation strategy*. The reduction of a set of equality literals to solved form with the unification rules only is just *Robinson unification* (J.A. Robinson 1965) and the combination of a resolution step followed by Robinson unification of the unification conditions is the common SLD-resolution. A strategy that applies only the resolution rule to the goals until we have a pure equality goal (not necessary solved), followed by a sequence of applications of the unification rules transforming the equality goal into a solved goal, is called a *totally lazy refutation strategy*. It is clear that an implementation of a totally lazy strategy will be rather inefficient, since a lot of backtracking will become necessary, and we will get the information of unification failures much to late. G. Huet (1972) uses a similar concept of refutation (constraint resolution) for higher order logics, where unification has similar problems as E-unification. Every strategy between these two extreme cases is called a *lazy refutation strategy*. We want to use the concept of lazy strategies to incorporate equational theories into logic programs.

Let $E := \{ s_1 = t_1,\ldots,s_n = t_n \}$ be a finite set of term pairs. An equational theory is the finest congruence relation on the term algebra $\mathcal{T}$ that contains all term pairs $\sigma s = \sigma t$ with $s = t \in E$ and with $\sigma \in SUB$. We denote this congruence by $=_E$ and call it E-equality. It is clear that the equality facts of a logic program induce such an equational theory. By a result of G. Plotkin (1972) for the whole first order predicate logics we can remove the equality facts and the equality clauses from a program, if we replace Robinson unification by E-unification under the standard strategy, where E is the removed set

of equality facts, and if no clauses with equality head remain in the program. We call this a standard E-refutation strategy. *E-unification* is the computation of substitutions $\sigma \in SUB$ for an equality goal $\Leftarrow s_1 = t_1,\ldots,s_n = t_n$ (also called *E-unification problem*), such that $\sigma s_i =_E \sigma t_i$ for each i $(1 \leq i \leq n)$. We write $U_E(\Gamma)$ to denote the set of all such *E-unifiers* of an equality goal Γ. The E-unifiers can be obtained as answer substitutions to the query $\Leftarrow s_1 = t_1,\ldots,s_n = t_n$ to a program consisting just of the equality facts of E and the equality clauses. This implies the soundness of Plotkin's method for first order Horn logics. J. Jaffar, J.-L. Lassez & M. Maher (1986) show the completeness of the standard E-refutation strategy for Horn logic with E-equality; see also (J.H. Gallier & S. Raatz 1986).

Theorem: *SLD-Resolution with E-unification is sound and complete for programs without functional clauses.* ■

A set $\mu U_E(\Gamma)$ of substitutions is called a *base* or a *minimal, complete* set of E-unifiers of Γ, iff

(i) $\mu U_E(\Gamma) \subseteq U_E(\Gamma)$ (correctness)

(ii) $\forall\, \delta \in U_E(\Gamma)\ \exists\, \sigma \in \mu U_E(\Gamma)$ with $\delta x =_E \lambda\sigma x$ (completeness)
(for all $x \in \mathcal{V}(\Gamma)$ and some $\lambda \in SUB$)

(iii) $\sigma, \tau \in \mu U_E(\Gamma)$ with $\sigma x =_E \lambda\tau x$ (for all $x \in \mathcal{V}(G)$ and some λ) (minimality)
implies $\sigma = \tau$

The elements of $\mu U_E(\Gamma)$ are called *most general E-unifiers* (E-mgu) of Γ.
An E-unification problem Γ is *unitary*, if it has a base with a single E-mgu, i.e., if $|\mu U_E(\Gamma)| \neq 1$.

Given a set D of distinguished equality goals, called *extended disagreements*, the application of the following transformation rules to an equality query is called *lazy unification with D*.

(T) $x = x \,\&\, G \longrightarrow G$ *(tautology)*

(B_L) $x = t \,\&\, G \longrightarrow x = t \,\&\, \{x \leftarrow t\}G$ *(lazy binding)*
if $x \notin \mathcal{V}(t)$, but $x \in \mathcal{V}(G)$ and $\{x \leftarrow t\}$ is a substitution of x by t
and if the goal $\Leftarrow x = t$ is in D

(D_L) $f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n) \,\&\, G \longrightarrow s_1 = t_1 \,\&\, \ldots \,\&\, s_n = t_n \,\&\, G$ *(lazy decomposition)*
provided the goal $\Leftarrow f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n)$ is not in D

(O) $t = x \,\&\, G \longrightarrow x = t \,\&\, G$ *(orientation)*

Usually D should contain all goals $\Leftarrow x = t$, such that the binding rule is not lazy. To obtain Robinson unification D must just consist of all goals $\Leftarrow s = t$, where s and t start with different symbols, since then the decomposition rule is always applicable for term pairs starting with the same function symbol. In this case D is just the set of disagreements as defined by J.A. Robinson (1965).

To get *lazy E-unification* we modify the lazy decomposition rule, such that it decomposes also all term pairs with *decomposable* top-symbols (C. Kirchner 1984), if we have these subgoals not in D. An equality literal $f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n)$ is E-decomposable, iff

$$U_E(\, f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n)\,) = U_E(\, s_1 = t_1\ ,\ldots,s_n = t_n\,).$$

We assume D to be the set of all E-unification problems with $|\mu U_E(\Gamma)| \neq 1$, and containing no E-decomposable goals. Provided there is an algorithm that computes a base for each unitary E-unification problem, we can add the following merging rule for a focused subgoal Γ

(M) $\Gamma \,\&\, G \longrightarrow x_1 = t_1 \,\&\, \ldots \,\&\, x_n = t_n \,\&\, G$ *(unitary merging)*
if $\Gamma \notin D$ and $\{x_1 \leftarrow t_1,\ldots,x_n \leftarrow t_n\}$ is the E-mgu of Γ

For an implementation of lazy E-unification we should add some failure rules that terminate rule application, whenever a subgoal is focussed that is not E-unifiable (H.-J. Bürckert 1986).

Now, we can combine these lazy E-unification rules with the resolution rule (R) to obtain a lazy E-refutation strategy. A lazy E-refutation of a query Q with a program P with built in E-equality is a sequence of applications of (R) and the lazy E-unification rules starting with Q and terminating with an E-unifiable equality goal Γ. Every E-unifier of Γ computed by a suitable E-unification algorithm is an *E-answer* to Q.

Theorem: *Lazy E-refutation is sound and complete for programs without functional clauses.*

Sketch of Proof: Soundness is obvious. Completeness follows immediately with the Switching Lemma (independence of focused goal literal, J.W. Lloyd 1984) applied to the program consisting of P, the equational facts E and the equality clauses. A refutation with this extended program can be rearreanged, such that resolution with clauses of E or with equality clauses will be delayed to the end. Hence the first part of the rearranged refutation is essentially a lazy E-refutation and the last part is E-unification. ■

Lazy E-refutation may be used to extend the Abstract Prolog Machine of D.H.D Warren (1983) for logic programs with certain built in equational theories (H.-J. Bürckert 1986). The main idea is to compile lazy E-unification into suitable instructions for the Warren Machine and to collect the extended disagreements, that occur in the goals during computation, in a special memory. After termination the remaining disagreements are E-unified. If they are not E-unifiable, backtracking will be used to try an alternative path.

Literature

H.-J. Bürckert: *Lazy Theory Unification in Prolog: An Extension of the Warren Abstract Machine*, Proc of GWAI'86, Springer, Informatik Fachberichte 124, 1986

F. Fages & G. Huet: *Complete Sets of Unifiers and Matchers in Equational Theories*. J. of Theoret. Comp. Sci., 1986

G. Huet: *Constraint Resolution: A Complete Method for Higher Order Logic*, Thesis, Case Western Reserve University, 1972

J.H. Gallier, S. Raatz: *SLD-Resolution Methods for Horn Clauses with Equality based on E-Unification*, Proc of Symp. on Logic Programming, Salt Lake City, 1986

J. Jaffar, J.-L. Lassez, M.J. Maher: *Logic Programming Language Scheme*, in *Logic Programming: Functions, Relations, Equations* (ed. D. DeGroot, G. Lindstrom), Prentice Hall, 1986

C. Kirchner: *A New Equational Unification Method: A Generalization of Martelli-Montanari's Algorithm*, Proc of CADE'84, Springer, LNCS, 1984

J.W. Lloyd: *Foundations of Logic Programming*, Springer, 1984

H. J. Ohlbach: *The Semantic Clause Graph Procedure - A First Overview*, Proc of GWAI'86, Springer, Informatik Fachberichte 124, 1986

J. Siekmann: *Unification Theory*, to appear in J. of Symb. Comp., 1987

D.H.D. Warren: *An Abstract Prolog Instruction Set*, SRI Technical Note, Stanford, 1983