Comments on Kornfeld's "Equality for Prolog": e-unification as a mechanism for augmenting the Prolog search strategy.

E. W. Elcock and P. Hoddinott

Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract

The search strategy of standard Prolog can lead to a situation in which a predicate has to be evaluated in circumstances where it has an infeasibly large number of instantiations. The work by Kornfeld [8] addressed this important problem by means of an extension of unification which allows Prolog to be augmented by what is essentially a (non-standard) equality theory.

This paper uses the notion of the general procedure introduced by van Emden and Lloyd [12] to formalize Kornfeld's work. In particular, the formalization is used to make a careful analysis and evaluation of Kornfeld's solution to the problem of delayed evaluation.

1. Introduction

The work by van Emden and Lloyd [12] shows how the notion of a *general procedure* augmented by a particular *equality theory* can be used to make overt the logical framework common to apparently quite different systems. In their paper they present an account of Prolog II [2], as essentially the general procedure together with an appropriate equality theory. As they remark, this is particularly interesting in that Colmerauer's own presentation of Prolog II is one in which Prolog II is regarded as a system for manipulating infinite trees, and presented as a complete departure from a system based on first order logic.

The paper by van Emden and Lloyd makes reference to other novel work which attempts to incorporate equality into Prolog programming. In particular they refer to the work by Kornfeld [8].

Kornfeld's work in this area is particularly provocative. As Goguen [6] points out, Kornfeld gives no theoretical justification for his approach, and that it is in fact incomplete - although Goguen gives no clarification of his criticism of incompleteness that would not apply to all feasible logic programming implementations. Nevertheless, the underlying notions of Kornfeld's work are intuitively appealing. In what follows, we use the method of van Emden and Lloyd to elaborate Goguen's remark, and at least attempt to expose what is hidden in Kornfeld's work.

Section 2 contains a brief reminder of the general procedure augmented by an equality theory. Section 3 briefly introduces Kornfeld's work, and section 4 shows how the central notion can be formalized in the framework of the general procedure. The remaining sections focus on Kornfeld's use of an equality theory as a mechanism for augmenting the Prolog search strategy to handle delayed evaluation of goal predicates.

2. The General Procedure

The following description of what van Emden and Lloyd call the *general procedure* is taken from [12].

Definition. The *homogeneous form* of a clause $p(t_1, \ldots, t_n) \leftarrow B$ is

$$p(x_1, \ldots, x_n) \leftarrow x_1 = t_1, \ldots, x_n = t_n, B$$

where $x_1, \ldots, x_n$ are distinct variables not appearing in the original clause.

Definition. Let P be a program. The homogeneous form P' of P is the collection of the homogenous forms of each of its clauses.

Definition. An atomic formula, whose predicate symbol is "=", is called an *equation.*

We now describe the *general procedure.* We call it "general" because, depending on the theory of equality invoked after it, we get Prolog, Prolog II, or other specialized languages.

The work was funded in part by NSERC grant A2445, and by IBM Corporation.

The general procedure uses the homogeneous form P' of P, and produces an SLD-derivation [9, 11]. It consists of constructing, from an initial goal G, an SLD-derivation using input clauses from P', while never selecting an equation. The general procedure terminates if a goal consisting solely of equations is reached. Note that because of the homogeneous form of P', the general procedure never constructs bindings for the variables in the initial goal.

For a particular language, the general procedure needs to be supplemented by a theory E of equality. E is used to prove the equations resulting from the general procedure. During the proving of the equations, substitutions for variables in the initial goal are produced. If the equation-solving process is successful (that is, the empty goal is eventually produced), then these substitutions for the variables in the initial goal are output as the answer.

Van Emden and Lloyd show that with an equality theory consisting solely of the reflexive axiom ($x = x$), the general procedure is equivalent to Prolog. To the extent that one adds other axioms of "equality", one creates other applied logics.

3. Kornfeld's Implementation of an Equality for Prolog

Kornfeld modified a Lisp embedded Prolog system [7] to make an intuitively appealing change in the behaviour of the interpreter on unification failure. Kornfeld's informal description of the change can be paraphrased as follows: If the interpreter attempts to unify two terms Φ and Ψ and fails, then the interpreter attempts to establish the goal $equals(\Phi,\Psi)$ where *equals* is a user defined predicate in the Prolog program. If this goal succeeds, the resulting bindings are added to the binding environment and the original unification is deemed to have succeeded. If the goal fails then the goal $equals(\Psi,\Phi)$ is tried. If this succeeds then the resulting bindings are added to the binding environment and the original unification is deemed to have succeeded: otherwise the original unification is deemed to have failed. We will call this mechanism *extended unification* (e-unification).

This informal description leaves much to be desired in the way of specificity: much of the operational semantics has to be induced from Kornfeld's examples. For this reason, in the remainder of this section we consider two of Kornfeld's examples to motivate our interpretation of what we perceive is the intended operational semantics of e-unification.

Consider the introduction of the concept of rationals as a quotient set of ordered pairs of integers. To do this we define an equivalence relation over ordered pairs. This equivalence relation can then be used to define equality over rationals. Kornfeld does this by introducing the axiom

$$E1rat: \quad equals(rat(XN,XD), rat(YN,YD)) \leftarrow times(XN,YD,Z), times(XD,YN,Z).$$

Consider now the first of Kornfeld's examples illustrating e-unification.

The terms $rat(2,3)$ and $rat(X,6)$ are said to be e-unifiable. We are told that (standard) unification is attempted first and fails. Kornfeld tells us that this failure results in e-unification generating $\leftarrow equals(rat(2,3), rat(X,6))$, and that this goal succeeds by $E1rat$, binding X to 4.

Let us now consider this example in more detail. Standard unification would attempt to unify $rat(2,3)$ with $rat(X,6)$ as follows: starting with the leftmost symbol of each term, the algorithm would find a disagreement at the third symbol and would attempt to unify 2 with X. This would succeed, binding X to 2. The next disagreement would lead to an attempt to unify 3 with 6, which would fail. Kornfeld's explanation does <u>not</u> give the impression that e-unification would now lead to a call of $equals(3,6)$, that is, a call to "equals" as a result of (standard) unification failure for a subterm.

We therefore assume that the e-unification of a goal $p(t_1, \ldots, t_n)$ with the head $p(s_1, \ldots, s_n)$ of a clause is successful *iff* for each i, $i \leq n$, the argument t_i is e-unifiable with the argument s_i. The e-unification of an argument t_i with an argument s_i is successful ***if*** t_i and s_i are unifiable, ***else if*** $equals(t_i,s_i)$ can be established, ***else if*** $equals(s_i,t_i)$ can be established.

Let us now consider Kornfeld's formalization of the concept that a rational is equal to an integer. Kornfeld formalizes the equivalence relation with the following non-logical axioms:

$$E2rat: \quad equals(rat(N,D),I) \leftarrow integer(I), var(N), var(D),$$

$$N = I,\ D = 1.$$

$E3rat$: $equals(rat(N,D),I) \leftarrow$
$integer(I),$
$times(D,I,N).$

The predication $integer(I)$ is a Prolog [1] evaluable predicate which is true just in case I is an integer, and fails otherwise. The predication $var(X)$ is a Prolog evaluable predicate which is true just in case X is an variable, and fails otherwise. The predication $t = s$ is a Prolog evaluable predicate which now is true just in case t and s are e-unifiable, and fails otherwise.

The following is Kornfeld's second example illustrating e-unification. Kornfeld says that in the presence of the axioms $E1rat$, $E2rat$ and $E3rat$, the goal

$\leftarrow mem(rat(4,X), [2,3,cons(Y,Z),rat(R,W),rat(2,7)])$

will succeed three times with respectively, X bound to 2, R bound to 4 and X bound to W, and X bound to 14.

Again let us consider this example in more detail. It suffices to simplify the example so that the goal, G, is $\leftarrow mem(rat(4,X),[2])$. As Kornfeld does not give an axiomatization of "*mem*" we assume that "*mem*" has the following standard Prolog axiomatization:

$Mem1$: $mem(A,[A|L]) \leftarrow$
$Mem2$: $mem(A,[B|L]) \leftarrow mem(A,L).$

If the goal G is to succeed it must be e-unifiable with axiom $Mem1$.

The e-unification of G with $Mem1$ proceeds as follows: The e-unifier first attempts to unify $rat(4,X)$ with A. The unification of $rat(4,X)$ with A succeeds with $rat(4,X)$ bound to A. Next, an attempt is made to unify [2] with $[rat(4,X)|L]$. This attempt fails, and by our earlier assumption the e-unifier now generates $\leftarrow equals([2],[rat(4,X)|L])$. By inspection of axioms, $E1rat$, $E2rat$ and $E3rat$, it is clear that this and the symmetrized version $\leftarrow equals([rat(4,X)|L],[2])$ both fail. Accordingly, G fails, apparently contradicting Kornfeld's claim about this example.

We can however remove the contradiction. Although Kornfeld nowhere discusses their presence or necessity in his system, problems like the above do not arise if we assume that, for every function symbol in the lexicon of the program, we introduce a function substitutivity axiom. In the example above, let us add the axiom

$E1list$: $equals([H1|T1],[H2|T2]) \leftarrow$

$$H1 = H2,\ T1 = T2.$$

It is easy to see that the goal $\leftarrow equals([2],[rat(4,X)|L])$ which failed in the previous analysis, now succeeds.

In what follows we will take the view that e-unification is formalized as assumed, but that where appropriate, equality theories that build on e-unification will include appropriate function substitutivity axioms.

4. The general procedure and e-unification.

The general procedure allows us to capture e-unification, as expressed above, using the equality theory

$E1$: $eq(X,X) \leftarrow$
$E2$: $eq(X,Y) \leftarrow equals(X,Y)$
$E3$: $eq(X,Y) \leftarrow equals(Y,X)$

$E1$ is the "Prolog" equality axiom providing standard unification. Axioms $E2$ and $E3$ provide the symmetric extension to Kornfeld's predicate "*equals*". Finally a particular equality theory built on e-unification would be completed and distinguished by the particular axiomatization of the predicate "equals". As observed at the end of the proceeding section, these axioms would typically include a selected set of axioms from the axiom schema for function substitutivity such as $E1list$ below.

To illustrate this we show how the last example discussed above is handled using the general procedure. The program would include the axiomatization of "*mem*" which, in homogeneous form, is

$Mem1$: $mem(X,L) \leftarrow eq(X,X1),\ eq(L,[X1|L1])$
$Mem2$: $mem(X,L) \leftarrow eq(X,X1),\ eq(L,[Y1|L1]),$
$mem(X1,L1).$

Along with $Mem1$ and $Mem2$ the program also includes the following equality theory:

$E1$: $eq(X,X) \leftarrow$
$E2$: $eq(X,Y) \leftarrow equals(X,Y)$
$E3$: $eq(X,Y) \leftarrow equals(Y,X)$

$E1list$: $equals([H1|T1],[H2|T2]) \leftarrow$
$eq(H1,H2),\ eq(T1,T2)$

$E1rat$: $equals(rat(XN,XD),rat(YN,YD)) \leftarrow$
$times(XN,YD,Z),$
$times(XD,YN,Z)$

$E2rat$: $equals(rat(N,D),I) \leftarrow$

$integer(I)$, $var(N)$, $var(D)$,
$eq(N,I)$, $eq(D,1)$

$E3rat$: $equals(rat(N,D),I) \leftarrow$
$integer(I)$,
$times(D,I,N)$.

The goal $\leftarrow mem(rat(4,X),[2])$ is reduced by $Mem1$ to

$\leftarrow eq(rat(4,X),X1),\ eq([2],[X1|L1])$.

The first equation succeeds by $E1$ leaving

$\leftarrow eq([2],[rat(4,X)|L1])$.

Using $E2$ this goal is reduced to

$\leftarrow equals([2],[rat(4,X)|L1])$

which, using $E1list$, reduces to

$\leftarrow eq(2,rat(4,X)),\ eq([],L1)$.

The first equation, using $E2$, results in the goal clause

$\leftarrow equals(2,rat(4,X)),\ eq([],L1)$.

The first equation in this goal clause fails, but its symmetrized version reduces to □ by $E3rat$. The second equation in this goal clause reduces to □ by $E1$. A fuller discussion on Kornfeld's approach to equality and in particular, its relation to classical equality, can be found in [3].

5. Incompleteness and "partially specified objects"

Possibly the most provocative part of Kornfeld's treatment of unification failure, and one on which he places considerable emphasis, is his attempt to use the binding environment to mitigate the problem of dealing with potentially very large (possibly infinite) sets of instantiations of variables in a goal predication. This is a problem that standard Prolog [1] attempts to deal with by "evaluable predicates" - a partial solution at best, and one which exacerbates the incompleteness of standard Prolog [4].

Kornfeld attempts to deal with the problem by arranging to delay the evaluation of such a goal predicate. The delayed evaluation notionally constrains the original uninstantiated variables in that predication. Kornfeld calls such constrained variables "partially specified objects" and represents them in the object language.

We begin with a brief and informal motivation for delayed evaluation. The general problem is sufficiently important from the point of view of pragmatic implementations however, that a detailed analysis and comment on Kornfeld's treatment is presented in the following sections.

Consider a typical Prolog goal evaluable predicate such as $times(X,Y,Z)$, called when at least two of the variables X, Y and Z are uninstantiated. Standard Prologs, sensibly enough, refuse to initiate an exploration of the infinite space of possible instantiations, and the call fails.

A better approach is that taken in Absys [5] where the unevaluated assertion is associated with each of the uninstantiated variables. The associated assertion is automatically reprocessed if and when any of its associating variables is instantiated. If processing ends with variables which still have sets of associated assertions, then the system uses these sets to give the premises under which the set of assertions could be valid. Thus, and very trivially, the assertions $gr(X,Y)$, $plus(2,3,Y)$ might generate "yes: if gr(X,5)".

This capability was achieved in Absys by appropriate design at the level of the interpreter primitives. Kornfeld obtains a similar effect by exploiting his treatment of unification failure and subsequent appeal to a special "equality" theory. In the following sections we will analyse Kornfeld's approach by casting his apparatus and his illustrative example in the framework of the general procedure and an equality theory.

6. Delayed evaluation using e-unification

We begin with an informal motivation of what Kornfeld refers to as a property of his "equality" theory - that of treating "partially instantiated data objects" which, in turn, stand for "non-singleton sets of the Herbrand Universe of the program."

Suppose we want to write the code for an evaluable predicate "*gr*" (greater-than) over pairs of integers. We want to take into account the possibility that the predication gets called when one or both of its arguments are not yet instantiated to integers. We do *not* want to respond by an arbitrary instantiation from the infinity of numbers available. Somehow we have to associate the unevaluated primitive predication "*gr*" with the uninstantiated variable or variables, with the intent that this association is to act as a constraint on any other attempted instantiations of that variable or variables. The only *Prolog* mechanism we have for forming such an association is the "binding"

environment (substitution composition). Kornfeld writes the code for the primitive evaluable predicate "*gr*" so that each uninstantiated variable is bound to what is called an "Ω-term" (i.e. a 2-ary term with functor "Ω"). The arguments of the Ω term are a newly created variable, and the original predication, but with the original variable replaced by the new variable. The new variable plays the role of a *surrogate* for the original variable. Kornfeld now relies on the fact that, if the system attempts to unify the original variable, now bound to an Ω-term, with some other term, then the unification fails and the "equality theory" is invoked. The equality axioms for Ω-terms are written so that the failed unification is repeated, but this time with the *surrogate* of the original variable - in effect mimicing "try the (original) goal again". Success allows the original computation to proceed.

The following is a simple illustrative example. Suppose we have the goal $\leftarrow gr(X,3), mem(X,[4,2])$. The call of the evaluable predicate "$gr(X,3)$" leads to X being bound to $\Omega(Xs, gr(Xs,3))$, where Xs is a new variable introduced as a surrogate for X, and "$gr(Xs,3)$" is a surrogate for the original predication involved. The "*mem*" predication is now called. The attempt to unify X and 4 fails because X is bound to the Ω-term. The unification failure now invokes an appeal to the equality theory which eventually leads to a call of $\leftarrow equals(4, \Omega(Xs,gr(Xs,3)))$. The equality axioms for Ω have the effect of binding Xs (the surrogate of X) to 4, and calling $gr(Xs,3)$ (the predicate denoted by the surrogate of $gr(X,3)$) with this binding, which call succeeds. The original attempt to "unify" X with 4 is now deemed to have succeeded and the evaluation of the goals proceeds.

In what follows we exhibit *a standard Prolog equality theory for* Ω which we think is faithful to Kornfeld's Lisp/Prolog implementation. We will use this formulation to analyse "delayed evaluation". We will show that Kornfeld's method as presented by him is both incomplete and unsound. We also show that, even within the subset of "successful" evaluations the mechanism is inefficient and present a more efficient alternative. We take the final position that the use of an "equality" theory is an interesting mechanism for addressing undesirable effects of the Prolog search strategy and merits further study.

7. An illustrative predicate

For the expository purposes of this paper, it will be sufficient to consider a single predicate *gr* with the intended interpretation that $gr(X,Y)$ asserts that X and Y denote integers, and that the number denoted by X is greater than the number denoted by Y. The clauses Kornfeld uses to define the predicate *gr* are:

$gr1$: $gr(M,N) \leftarrow$
$\quad instantiated(M,M1),$
$\quad instantiated(N,N1),$
$\quad M1 > N1.$

$gr2$: $gr(M,N) \leftarrow$
$\quad instantiated(M,M1),$
$\quad eq(N,\Omega(Ns,gr(M1,Ns))).$

$gr3$: $gr(M,N) \leftarrow$
$\quad instantiated(N,N1),$
$\quad eq(M,\Omega(Ms,gr(Ms,N1))).$

$gr4$: $gr(M,N) \leftarrow$
$\quad eq(N,\Omega(Ns,gr(Ms,Ns))),$
$\quad eq(M,\Omega(Ms,gr(Ms,Ns))).$

Suppose that, at the time $gr(X,Y)$ is called, X and Y are indeed instantiated by integers n and m respectively, then the call of *gr* can simply be replaced by the Prolog system predicate "$>$". This motivates the first clause. The other three clauses are intended to capture those cases in which, at the time $gr(X,Y)$ is called, one or both of the arguments are not currently instantiated by integers. The reader can assume that the predicate "*instantiated*" has the expected interpretation: its formal definition will be deferred until after we have exhibited the domain of its potential arguments.

Consider the goal clause $\leftarrow gr(X,4),eq(X,5)$. The predication $gr(X,4)$ reduces by $gr3$ to $\leftarrow eq(X, \Omega(Xs,gr(Xs,4)))$ which in turn is reduced by the axiom $eq(X,X)$, (the first axiom of the equality theory to be developed), to $\square$ with the binding $X/\Omega(Xs,gr(Xs,4))$. We are left with the goal $\leftarrow eq(\Omega(Xs,gr(Xs,4)),5)$. It is clear that our intended Ω-equality theory should bind "Xs", (the surrogate for "X") to "5" and verify the surrogate "$gr(Xs,4)$" for the original relation "$gr(X,4)$".

Indeed, a first (oversimplified) Ω-equality theory would have to include the following clauses:

$E1$: $eq(X,X) \leftarrow$
$E2$: $eq(X,Y) \leftarrow equals(X,Y)$
$E3$: $eq(X,Y) \leftarrow equals(Y,X)$

$E4$: $equals(\Omega(Vsurrogate,Psurrogate),Thing) \leftarrow$
$instantiated(Thing,Value),$
$eq(Vsurrogate,Value),$
$Psurrogate.$

Clause $E4$ is certainly adequate for the illustrative example above: the recursive appeal to the full Ω-equality theory in "$eq(Vsurrogate,Value)$" would succeed trivially, in this simple example, by $E1$, when "$Psurrogate$" would succeed. The final bindings for the example are $X/\Omega(Xs,gr(Xs,4)$, $Xs/5$. $\Omega(5,gr(5,4))$ is the instance of the Ω term bound to X. Kornfeld interprets such a ground instance as an alternative representation of the constant 5.

Consider now the slightly more complex goal clause $\leftarrow gr(X,4),gr(Y,5),eq(X,Y)$. As before, the first two predications succeed with the bindings $X/\Omega(Xs,gr(Xs,4))$ and $Y/\Omega(Ys,gr(Ys,5))$. The evaluation of the remaining goal $eq(X,Y)$ now introduces and raises the question as to how the Ω-equality theory should deal with equality between two Ω-terms. Clearly, the predication "$eq(X,Y)$" asserts that X and Y denote the same individual. This, in turn, implies that the conjunction of any separate constraints on X and Y must constrain this individual. In addition then to asserting that the surrogates for X and Y are equal, we have to arrange to introduce a new Ω-term whose Vsurrogate is equal to the Vsurrogates for X and Y and whose Psurrogate is the conjunction of the Psurrogates for X and Y.

We need to extend the Ω-equality theory by the clause

$E5$: $equals(\Omega(Vs1,Ps1),\Omega(Vs2,Ps2)) \leftarrow$
$not\_instantiated(Vs1),$
$not\_instantiated(Vs2),$
$conjoin\_relations(Vs1,Ps1,Vs2,Ps2,Vs,Ps),$
$eq(Vs1,Vs2),$
$eq(Vs2,\Omega(Vs,P)).$

The above axioms $E4$ and $E5$ are as given by Kornfeld in [8].

When we come to define the predicate "*instantiated*" we will see that the first two predications in the body of the clause are intended to ensure that we are indeed dealing with the general case, i.e. where neither Ω argument of the head of the clause has a fully constrained substitution instance in the current binding environment, (i.e. neither argument denotes a known individual).

The definition of the predicate "*conjoin_relations*" is a straightforward piece of Prolog non-logical wizardry! It is:

```
conjoin_relations(X1,R1,X2,R2,X,R) ←
        replace_occurrence(X1,R1,X,NR1),
        replace_occurrence(X2,R2,X,NR2),
        conjoin(NR1,NR2,R).

replace_occurrence(T,(R,Rs),NT,(NR,NRs)) ←
        R =.. [F|Args],
        replace(T,Args,NT,NArgs),
        NR =.. [F|NArgs],
        replace_occurrence(T,Rs,NT,NRs).

replace_occurrence(T,R,NT,NR) ←
        R =.. [F|Args],
        replace(T,Args,NT,NArgs),
        NR =.. [F|NArgs].

replace(_,[],_,[]).

replace(T,[Arg|RArgs],NT,[NT|NRArgs]) ←
        T == Arg,
        replace(T,RArgs,NT,NRArgs).

replace(T,[Arg|RArgs],NT,[NT|NRArgs]) ←
        replace(T,RArgs,NT,NRArgs).

conjoin((A,B),R,(A,RR)) ←
        conjoin(B,R,RR).

conjoin(A,R,(A,R)).
```

The =.. is a Dec-10 Prolog evaluable predicate with, for example, $f(A,B,C)$ =.. $[f,A,B,C]$ being a true goal. The == is also a Dec-10 Prolog evaluable predicate which is true only if its two arguments are syntactically identical (i.e. even the variable names must be the same).

For our example goal clause

$\leftarrow gr(X,4),gr(Y,5),eq(X,Y)$

the axiom $E4$ invoked by the predication "$eq(X,Y)$" results in the bindings

Xs/Ys and $Ys/\Omega(XYs,(gr(XYs,4),gr(XYs,5)))$.

In effect, the binding for X, is now

$\Omega(\Omega(XYs,(gr(XYs,4),gr(XYs,5))),$
$gr(\Omega(XYs,(gr(XYs,4),gr(XYs,5))),4))$

The substitution instance of X is an Ω-term whose Vsurrogate is itself an Ω-term. In general, this nesting, involving binding Ω-terms to the surrogates of Ω-terms, may have arbitrary depth. We will call such a nesting a surrogate chain. The *first* Ω term in a surrogate chain is that Ω term which is bound to a program

variable. The *last* Ω term in a surrogate chain is that Ω term whose surrogate is not bound to an Ω term.

In the example, we see that the "last" Ω-term contains the Psurrogate, "$gr(XYs,4),gr(XYs,5)$" that expresses the complete constraint on X. The residual content of the full Ω-term bound to X is redundant. It should be apparent that this will always be so.

Finally, consider the goal clause

$\leftarrow gr(X,4),gr(Y,5),eq(X,Y),eq(X,6)$

which is identical to the initial goal clause in the previous example except for the additional goal $eq(X,6)$. In accordance with the previous example, the goals $gr(X,4),gr(Y,5),eq(X,Y)$ will all succeed. It remains to establish the goal

$$\leftarrow eq(\Omega(\Omega(XYs,(gr(XYs,4),gr(XYs,5))),\\ gr(\Omega(XYs,(gr(XYs,4),gr(XYs,5))),4)),\ 6).$$

This goal will succeed by $E3$ and $E4$, and will result in the program variables X and Y being bound to the completely specified object

$$\Omega(\Omega(6,(gr(6,4),gr(6,5))),gr(\Omega(6,(gr(6,4),gr(6,5))),4))$$

which, following Kornfeld, we take as an alternative denotation of the number 6.

In general, for an Ω-term $\Omega(\Phi,\Psi)$ bound to a program variable X, the object which we will interpret as the denotation of X, will always be the last Ω-term in the Vsurrogate chain of $\Omega(\Phi,\Psi)$. If the Vsurrogate of this last Ω-term is a non-variable then the object identified by X is taken to be completely specified. In the sense of the predicate "*instantiated*" still to be defined, the non-variable instance of the surrogate of the last Ω-term is regarded as the value of the "instantiated" variable X. Otherwise, X is regarded as not instantiated.

With these considerations in mind, the formal definition of the predicate "*instantiated*" is:

$$\begin{aligned}&instantiated(Omega,Value) \leftarrow\\ &\qquad omega\_term(Omega,\Omega(X,\_)),!\\ &\qquad nonvar(X),\\ &\qquad var(Value),\\ &\qquad X{=}Value.\end{aligned}$$

$$\begin{aligned}&instantiated(Thing,Value) \leftarrow\\ &\qquad nonvar(Thing),\\ &\qquad var(Value),\\ &\qquad Thing{=}Value.\end{aligned}$$

The predicate "*omega_term*" has the intended interpretation that $omega\_term(X,Y)$ asserts that X is an omega-term and that Y is that last Vsurrogate omega-term in X in the sense discussed above. The formal definition of "*omega_term*" is:

$$\begin{aligned}&omega\_term(\Omega(X,R),Y) \leftarrow\\ &\qquad nonvar(X),\\ &\qquad omega\_term(X,Y).\\ &omega\_term(\Omega(X,R),\Omega(X,R)).\end{aligned}$$

8. Pragmatics

In the last section we motivated and explicated, in a standard Prolog, Kornfeld's Ω-equality theory. We will show that this mechanism for handling delayed evaluation is inefficient and we will show how some of the inefficiency can be removed whilst staying within the same conceptual framework.

We will first exhibit the source of the "inefficiency" by a (very) simple example. Consider the goal clause

$\leftarrow gr(X,Y), eq(X,3), eq(Y,2)$

Reduction of the leftmost predication gives

$$\leftarrow eq(Y,\Omega(Ys,gr(Xs,Ys))),\ eq(X,\Omega(Xs,gr(Xs,Ys))),\\ eq(X,3),\ eq(Y,2)$$

where Xs and Ys are surrogates for X and Y. It should be noted that we have the surrogate predication $gr(Xs,Ys)$ occurring *twice*, once in each Ω-term. Using $E1$ the two leftmost predications give rise the bindings

$Y/\Omega(Ys,gr(Xs,Ys)$ and $X/\Omega(Xs,gr(Xs,Ys)$

when we can write the current goal clause as

$\leftarrow eq(\Omega(Xs,gr(Xs,Ys)),3),\ eq(\Omega(Ys,gr(Xs,Ys)),2).$

Using $E2$ followed by $E4$, the leftmost predication is reduced to

$\leftarrow instantiated(3,Value),\ eq(Xs,Value),\ gr(Xs,Ys).$

The first predication binds *Value* to 3 when, using $E1$, the second predication binds Xs to 3, leaving the current goal clause

$\leftarrow gr(3,Ys),\ eq(\Omega(Ys,gr(3,Ys)),2).$

The leftmost predication is essentially "try again (the surrogate for) $gr(X,Y)$ but in the new binding context". This is certainly well motivated, but again note the replication of the predication $gr(3,Ys)$ - the instance of the replicated surrogate $gr(Xs,Ys)$ of the "original" predication $gr(X,Y)$.

Unfortunately the call of $gr(3,Ys)$ will be delayed. Unlike the Absys mechanism mentioned earlier, in order to delay we have to go through another level of surrogates and Ω-terms.

The leftmost predicate $gr(3,Ys)$ reduces by $gr2$ to

$\leftarrow instantiated(3,V),\ eq(Ys,\Omega(Yss,gr(V,Yss)))$

The call of instantiated binds V to 3, when $E1$ binds Ys to $\Omega(Yss,gr(3,Yss))$. The current goal clause is now:

$\leftarrow eq(\Omega(\Omega(Yss,gr(3,Yss)),gr(3,\Omega(Yss,gr(3,Yss)))),2).$

Axiom $E2$ replaces eq by $equals$ when $E4$ reduces the predication to

$\leftarrow instantiated(2,V1),\ eq(\Omega(Yss,gr(3,Yss)),V1),$
$\quad gr(3,\Omega(Yss,gr(3,Yss)))$

which, in turn, reduces to

$\leftarrow eq(\Omega(Yss,gr(3,Yss)),2),\ gr(3,\Omega(Yss,gr(3,Yss))).$

The leftmost predicate reduces by $E2$ and $E5$ to

$\leftarrow instantiated(2,V2),\ eq(Ysss,V2),\ gr(3,Ysss)$

all of which clearly reduce to $\square$.

It would be nice to stop here! However, we are still left with the "redundant" goal

$\leftarrow gr(3,\Omega(2,gr(3,2)))$

which is reduced by $gr1$ to

$\leftarrow instantiated(3,V3),\ instantiated(\Omega(2,gr(3,2))),V4),$
$V3 > V4.$

The first predication binds $V3$ to 3; the second determines that the Ω-term is indeed instantiated, (its last surrogate is "2"), and binds $V4$ to 2, when the call of $3 > 2$ succeeds.

Although, in the simple illustrative example considered here, unnecessary elaborations of surrogate chains and replication of surrogates predicates, are not traumatic, they can quickly begin to be so as the complexity of the original goal increases.

Kornfeld's equality axioms can be modified to avoid these unpleasant effects. The modification is to take advantage of what we know about the significance of a nesting structure for Ω-terms, and, as in the predicate "instantiated", deal directly and appropriately with the last Ω-term in any nesting.

We replace the equality axiom $E4$ by:

$E4$: $\quad eq(Omega,Thing) \leftarrow$
$\quad omega\_term(Omega,\Omega(X,R)),$
$\quad instantiated(Thing,Value),!,$
$\quad eq(X,Value),$
$\quad R,!.$

If we now reduce the goal

$\leftarrow eq(\Omega(\Omega(Yss,gr(3,Yss)),gr(3,\Omega(Yss,gr(3,Yss)))),2)$

using the new equality theory, we have:

$\leftarrow omega\_term($
$\quad \Omega(\Omega(Yss,gr(3,Yss)),gr(3,\Omega(Yss,gr(3,Yss)))),$
$\quad \Omega(Yns,Pns)),$
$\quad instantiated(2,V),\ eq(Yns,V),\ Pns,!$

The first predicate binds the new variable surrogate Yns to Yss, and the new predicate surrogate Pns to just $gr(3,Yss)$. The predicate $instantiated(2,V)$ now binds 2 to V, when $E1$ leads to V being bound to Yns. The call of Pns now becomes the straightforward predication $gr(3,2),!$ which succeeds.

This elegant change in the equality theory completely eliminates processing of irrelevant Ω-term structure.

Finally, in this section on pragmatics, let us consider an example in which the reduction of the goal leads to a conjunction. Consider the goal

$\leftarrow gr(X,4),gr(Y,5),eq(X,Y),gr(W,8),gr(V,9),$
$\quad eq(W,V),eq(X,W),eq(X,10).$

Reduction of the first three predications will lead to

$\leftarrow eq(\Omega(Xs,gr(Xs,4)),\ \Omega(Ys,gr(Ys,5)))$

with X and Y bound to the two Ω-terms respectively. Reduction using $E5$ which, in turn, uses "$conjoin\_relations$", succeeds with Xs and Ys bound to

$\Omega(XYs,\ (gr(XYs,4),gr(XYs,5)))$

and hence with X bound to

$\Omega(\Omega(XYs,(gr(XYs,4),\ gr(XYs,5))),$
$\quad gr(\Omega(XYs,(gr(XYs,4),gr(XYs,5))),\ 4))$

Similarly, the reduction of the next three predicates leads to a sequence of substitutions which bind W to

$\Omega(\Omega(XYs,(gr(XYs,8),gr(XYs,9))),$
$\quad gr(\Omega(XYs,(gr(XYs,8),gr(XYs,9))),8))$

and with the current goal

$\leftarrow eq(X,W),eq(X,10).$

The leftmost predication is in effect

$eq($
$\quad \Omega(\Omega(XYs,(gr(XYs,4),gr(XYs,5))),$
$\quad gr(\Omega(XYs,(gr(XYs,4,gr(XYs,5))),4))$
$\quad \Omega(\Omega(XYs,(gr(XYs,8),gr(XYs,9))),$
$\quad gr(\Omega(XYs,(gr(XYs,8,gr(XYs,9))),8))$
$\quad)$

This is reduced by $E5$, again involving a new Vsurrogate and a conjunction of predications, to $\square$ with X bound to W bound to

$\Omega(\quad \Omega(Z,(gr(Z,4),gr(Z,5),gr(Z,8),gr(Z,9),$
$\qquad gr(Z,8),gr(Z,9),gr(Z,4),gr(Z,8))),$
$\quad (gr(\Omega(Z,(gr(Z,4),gr(Z,5),gr(Z,8),gr(Z,9),$

$gr(Z,8),gr(Z,9),gr(Z,4),gr(Z,8))),4),$
$gr(\Omega(Z,(gr(Z,4),gr(Z,5),gr(Z,8),gr(Z,9),$
$gr(Z,8),gr(Z,9),gr(Z,4),gr(Z,8))),5),$
$gr(\Omega(Z,(gr(Z,4),gr(Z,5),gr(Z,8),gr(Z,9),$
$gr(Z,8),gr(Z,9),gr(Z,4),gr(Z,8))),8),$
$gr(\Omega(Z,(gr(Z,4),gr(Z,5),gr(Z,8),gr(Z,9),$
$gr(Z,8),gr(Z,9),gr(Z,4),gr(Z,8))),9)))$

Because of structure sharing, the size of these Ω-terms is not all that alarming. However this complexity of Ω-structure produced by Kornfeld's axiom $E5$ is unnecessary. We modify $E5$ (as we did $E4$) to ignore irrelevant Ω-structure.

In the same spirit as the earlier changes, we replace $E5$ by

$E5$: $eq(Omega1,Omega2) \leftarrow$
$omega\_term(Omega1,\Omega(X1,R1)),$
$var(X1),$
$omega\_term(Omega2,\Omega(X2,R2)),$
$var(X2),$
$conjoin\_relations(X1,R1,X2,R2,X,R),$
$X1=X2,$
$X2=\Omega(X,R).$

The equality between Ω-terms above now reduces by the new $E5$ to

$\leftarrow conjoin\_relations(XYs,(gr(XYs,4),gr(XYs,5)),$
$WVs,(gr(WVs,8),gr(WVs,9)),XWs,PXws),$
$XYs=WVs, WVs=\Omega(XWs,PXWs)$

and we end up with X bound to Xs to XYs to WVs to

$\Omega(XWs,(gr(XWs,4),gr(XWs,5),gr(XWs,8),gr(XWs,9)))$

where all unnecessary structure has been eliminated.

9. Soundness and Completeness

Consider the goal

$\leftarrow gr(X,Y), eq(X,Y)$

which should fail. The first predication binds X and Y to the Ω-terms $\Omega(Xs,gr(Xs,Ys))$ and $\Omega(Ys,gr(Xs,Ys))$ respectively. The current goal is now

$\leftarrow eq(\Omega(Xs,gr(Xs,Ys)), \Omega(Ys,gr(Xs,Ys)))$

which succeeds by $E1$ with Xs bound to Ys. The inference system is unsound.

Consider the goal

$\leftarrow gr(X,Y), gr(X,3), eq(Y,5).$

The leftmost predication succeeds with the bindings $X/\Omega(Xs,gr(Xs,Ys))$ and $Y/\Omega(Ys,gr(Xs,Ys))$ respectively. We are left with the goal

$\leftarrow gr(\Omega(Xs,gr(Xs,Ys)),3), eq(\Omega(Ys,gr(Xs,Ys)),5).$

Using the axiom

$gr(M,N) \leftarrow$
$instantiated(N,N1),$
$eq(M,\Omega(Ms,gr(Ms,N1)))$

the leftmost predication in the goal reduces to

$\leftarrow eq(\Omega(Xs,gr(Xs,Ys)), \Omega(Xss,gr(Xss,3))).$

This reduces to $\square$ by $E1$ with Xs bound to Xss and Ys bound to 3. We are left with the remaining goal

$\leftarrow eq(\Omega(3,gr(Xss,3)),5)$

which reduces to

$\leftarrow eq(3,5), gr(Xss,3)$

which fails.

If the original goal were reordered to read

$\leftarrow eq(Y,5), eq(X,3), gr(X,5)$

the goal would succeed. This result would seem to defeat the motivation for introducing Ω-terms!

10. A wider context

In section 2 we introduced the notion of the generalized procedure based on the transformation of a Prolog program P to its homogeneous form P', supplemented by an (equality) theory of the predicate "=" of the homogeneous form.

The discussion of the Ω-terms in sections 5 to 9 above has made no appeal to the homogeneous form and the full general procedure as such. This is simply because all the relevant issues lie in the equality theory of the Ω-terms as such, and could be exposed using a very simple predicate "*gr*" whose conversion to homogeneous form would have added nothing to the exposition.

However, if the equality theory for Ω-terms were, as would be typical, embedded in a more comprehensive equality theory, then, as should be clear from sections 3 and 4, the full power of the complete equality theory is only realized in the context of the general procedure.

A simple illustrative example in the style of section 3 is the goal clause

$\leftarrow gr(X,rat(2,3)), mem(X,[rat(4,6)]).$

We assume that "*gr*" is a predicate over pairs of rationals specified in a way similar to the predicate in section 7, and that as in section 3 "*mem*" and "*rat*" have their usual axiomatization, and that the equality theory is augmented to deal with rationals and lists.

It is left as an exercise to the reader to show that this goal clause is reduced to □ only in the context of the general procedure.

11. Summary and conclusion

A paper by Kornfeld [8] presenting a notion of "extended unification" has received some attention and has been widely cited. The notion of extended unification is informally presented in the original paper and its intended operational semantics has to be largely induced from the examples given in the paper.

In this paper and in [3], we have attempted to give a more formal and clearer treatment, faithful to our perception of the original intuitions, but in the context of the general procedure [12].

In this paper we have focused our attention on Kornfeld's use of extended unification to appropriately delay evaluation of predicates with a large or possibly infinite set of instantiations - an important and interesting problem for Prolog implementations. We have formalized Kornfeld's basic notions of what we call the Ω-theory. The formalization is presented as an executable program written entirely in standard Prolog. Because of the unusual nature of this approach to delayed evaluation, we have given the Ω-theory in full.

We use the formalization to show that Kornfeld's method is potentially incomplete and unsound in a way that runs completely counter to its motivation. We have shown that the method is inefficient even after a nice modification.

Kornfeld's work does however provide interesting insights into the possibilities offered by "non-standard" equality theories, insights which we hope the present paper sharpens.

Our current work indicates that the difficulties identified in the Ω-theory can be resolved by a modified axiomatization. However, we take the position that it is not yet clear whether such a modified Ω-theory would have other than conceptual advantages over more direct methods, in which the necessary associations between variables and sets of, as yet unevaluated constraining predicates, are handled directly by the interpreter as in [5, 10].

References

1. Clocksin, W. F. and Mellish, C. S. *Programming in Prolog.* Springer-Verlag, New York, 1981.

2. Colmerauer, A. et. al. Prolog II: Reference Manual and Theoretical Model. Groupe d'Intelligence Artificielle, Faculte des Sciences deluming, Marseilles, 1982.

3. Elcock, E. W. and Hoddinott, P. Classical Equality and Prolog. TR 143, Department of Computer Science, The University of Western Ontario, 1985.

4. Elcock, E. W. The Pragmatics of Prolog: Some Comments. Proceedings Logic Programming Workshop'83, Portugal, June, 1983, pp. 94-106.

5. Foster, J. M. and Elcock, E. W. Absys 1: an incremental compiler for assertions - an introduction. In *Machine Intelligence*, Edinburgh University Press, Edinburgh, 1969, pp. 423-429.

6. Goguen, Joseph A. and Meseguer, J. "Equality, Types, Modules, and (Why Not?) Generics for Logic Programming". *The Journal of Logic Programming 1*, 2 (August 1984), 179-209.

7. Kahn, K. Unique Features of LM-Prolog. Unpublished manuscript.

8. Kornfeld, W. A. Equality for Prolog. Proceedings, Seventh International Joint Conference on Artificial Intelligence, 1983, pp. 514-519.

9. Kowalski, R. A. Predicate Logic as Programming Language. Proceedings, Information Processing, Amsterdam, 1974, pp. 570-574.

10. Naish, L. Automatic generation of control for logic programs. Technical Report 83/6, Dept. of Computer Science, University of Melbourne, 1983.

11. van Emden, M. H. Programming with Resolution Logic. In *Machine Intelligence 8*, Ellis Horwood, 1977, pp. 266-299.

12. van Emden, M. H. and Lloyd, J. W. "A Logical Reconstruction of Prolog II". *The Journal of Logic Programming 1*, 2 (August 1984), 143-150.