UCG-E: An Equational Logic Programming System

Lutz H. Hamel
BKS Software GmbH, Guerickestr. 27, 1000 Berlin 10, Germany
usenet: lhh@bksbln

UCG-E (Hamel, 1991) is an equational logic programming system inspired by O'Donnell's system (O'Donnell, 1985) and was designed to allow the integration of logic systems into imperative style programming environments. Equational logic systems are interesting, since they may be implemented very efficiently with term rewriting systems (Herman, 1991). Backtracking, a major source of inefficiencies in other logic systems, is unnecessary in equational logic if the confluency of the underlying rewriting system is guaranteed. O'Donnell has identified four restrictions to the form of the equational axioms which assure the confluency of a system. Since the restrictions are purely syntactical, no runtime checks need to be performed to see whether a transformation is legal or not.

The UCG-E system takes an equational specification and translates it into C code. The generated code consists of a lookup table and a set of C function definitions. Each specification has two parts: First, a *declaration section* containing the alphabet and variable declarations; second, an equational or *rule section*.

Each equation in the rule section has the form `M:=N` and represents a directed rewrite rule `M`→`N`, where the term `N` replaces the term matched by `M`. Currently the system enforces two restrictions on the form of the rewrite rules. Let `Q` be a well-formed term and `V(Q)` be the set of variables in term `Q`, then

(1) The term `M` must be linear.

(2) `V(N)` ⊆ `V(M)`.

The first restriction insures that if a rule matches an input term the variables in `M` are uniquely instantiated. The second restriction insures that instead of having to do a global unification of the variables we may copy the values of the variables from the left hand side to the right hand side during rewriting if necessary. As it turns out, these are two of the four restrictions which O'Donnell identified. The other two restrictions are currently not checked for, but have to be enforced by the programmer through disciplined programming.

Three features distinguish the UCG-E system from other equational logic systems:

User Defined Attributes -
: UCG-E allows the user to define and attach attributes to terms (especially, attributes of C-basic and C-typedef types).

Term Generating Functions -
: A term generating function is built for each symbol in the alphabet. The user may call these functions to build terms labeled by symbols from the alphabet. In addition of generating a term these functions make the new term known to the term rewriting

mechanism which attempts to apply any of the given rules to this newly generated term.

User Action Functions -

The user action functions allow the use of side effects in the rewrite rules. These side effects could take on the form of I/O or any other action which lies outside the realm of efficient equational processing. User action functions are distinguished symbols in the alphabet definition of the specification and may only appear on the right hand side of the rewrite rules.

Since the UCG-E system does allow side effects to be used in its rewriting rules, the rewrite sequence is as much part of the solution as is the final answer and therefore a deterministic selection of the rewrite sequence is important. UCG-E selects rules based on the order in which they appear in the specification rather than selecting them non-deterministically.

The programming and debugging of equational specifications is greatly facilitated by the integrated interactive debugger. The debugger allows the user to single-step rewrite sequences one rule at a time and to browse current state information.

As an illustration of a typical specification we show a program which appends an item to a list of items; in this case the items are integers.

```
%nullary nil;
%nullary item [int i];   /* 'item' has 'i' as an attribute */
%binary  cons;
%binary  append;

%var TERM head;
%var TERM tail;
%var int int_val;
%%
append(item(int_val), nil) := cons(item(int_val), nil);
append(item(int_val), cons(head, tail)) :=
       cons(head, append(item(int_val), tail));
```

References

Hamel, 1991.

Hamel, Lutz H., ''UCG-E Language Definition and User's Guide,'' *Technical Report*, BKS Software GmbH, (1991).

Herman, 1991.

Herman, M., Kirchner, C., and Kirchner, H., ''Implementations of Term Rewriting Systems,'' *The Computer Journal* **34** pp. 20-33 (January 1991).

O'Donnell, 1985.

O'Donnell, M., *Equational Logic as a Programming Language*, The MIT Press, Cambridge, Massachusetts (1985).