RAL-86-010

Science and Engineering Research Council
Rutherford Appleton Laboratory
CHILTON, DIDCOT, OXON, OX11 0QX

RAL-86-010

ERIL - Equational Reasoning: an Interactive Laboratory

A J J Dick

March 1985

ERIL - Equational Reasoning: an Interactive Laboratory

A. J. J. Dick

Science and Engineering Research Council
Rutherford Appleton Laboratory
Informatics Division
Software Engineering Group
Chilton, Didcot, OXON OX11 0QX, U.K.

March 1985

ABSTRACT

ERIL is an experimental laboratory for equational reasoning based on Knuth-Bendix superposition, designed specifically with two features in mind:-

1) The dynamic configuration of completion/proof strategies. ERIL supports a variety of types of equation, including rewrite-rules, undirected equations and hypotheses, whose functions are controlled through attribute lists which describe how sets inter-relate.

2) The treatment of certain classes of partial algebra by use of lattice-structured typing. This method overcomes certain problems that would otherwise require the use of conditional axioms.

Various forms of the Knuth-Bendix algorithm fall naturally into the framework presented by ERIL, and it is possible to tailor the system to perform completion modulo a set of equations, inductive proofs, program transformations, priority rewriting and other useful applications. Examples are given here of completion of a simple partial algebra, a proof that a particular group is commutative, and completion modulo a set of equations.

1. *INTRODUCTION*

ERIL is an experimental interactive laboratory for equational reasoning based on Knuth-Bendix superposition. A number of interactive facilities exist for the processing of rewrite-rules (e.g. REVE [Lescanne 83, Forgaard and Guttag 84], RRL [Kapur and Sivakumar 84]). Their emphasis has been on the generation of confluent sets of rewrite rules. ERIL is a tool for investigating a wide range of strategies, not only for the completion process, but for a number of other equational reasoning related applications.

A feature of ERIL is the ability to dynamically alter the attributes of the system. A variety of types of equation are permitted, including rewrite-rules, undirected equations and hypotheses to be proved. The whole equation base is divided into sets, whose number and function are specified by the user. The relationship between equations or rules within a set or between sets of rules can be dynamically altered. The user may supply orderings to be associated with particular sets of directed rules, as well as algorithms for sorting equations within a set. Various forms of the Knuth-Bendix algorithm fall naturally into this framework, including completion modulo a set of equations [Huet 80]. It is also possible to tailor the system to perform inductive proofs [Huet and Hullot 82], program transformations, priority rewriting [Baeten et al. 84], and a number of other useful applications.

Underlying ERIL is a lattice-structured typing method developed from related work by [Walther 83,84] and [Shamir and Wadge 77], which allows the treatment of certain classes of partial algebra, common in many applications, that would otherwise require the use of conditional rewrite-rules and a proliferation of error axioms. The undefined type is used to describe the result of applying a function to an argument outside its domain of definition, and superpositions that yield the undefined type are considered as meaningless. This technique partly overcomes a limitation of the original completion algorithm noted by the original authors [Knuth and Bendix 70 (example 18)] exemplified by the superposition of the axioms

$$inv(a).a = 1 \qquad \text{(i)}$$

$$a.0 = 0 \qquad \text{(ii)}$$

which produces the critical pair <0,1>, causing the equational system to degenerate. By defining the inverse function as partial, and in particular undefined on 0, ERIL is able to reject the above superposition as invalid, thus maintaining the consistency of the equational theory.

Section 2 describes the structure of ERIL, section 3 its procedures, and examples are given in section 4 of using ERIL to complete a set of axioms describing a partial algebra, to prove the commutativity of a particular group, and for completion modulo a set of equations.

ERIL forms part of the algebraic tools developed in Waterloo Prolog [Roberts] for the AVER project in the Department of Computing, Imperial College. Sponsored by SERC grant B85838, this project is aimed at providing an environment convenient for the specification and verification of computer programs.

2. *ERIL'S DATA-BASE*

At the lowest level, ERIL maintains a data-base of algebra definitions including types and functions. Dependent on this is the data-base of equalities and associated attributes; equalities are pairs of expressions structured according to the rules of the algebra. At the top level, the equalities are divided into sets with set attributes. In the following sub-sections, each of these levels is considered in detail.

2.1. ALGEBRA DEFINITIONS

2.1.1. *Types*

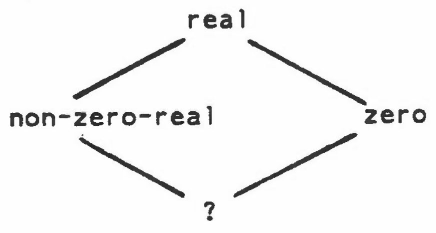
ERIL supports a lattice-structured type hierarchy which requires the user to give a partial ordering on types. There is by default a bottom type, '?', representing the undefined type, which closes the bottom of the lattice. If necessary, a top type, 'T', is inserted to close the top of the lattice. A type definition is given as a set of simple ordering statements. Example 1 shows two type orderings for the domains of i) the real numbers and ii) stacks of elements.

Example 1. Type definitions for the reals and stacks of elements.

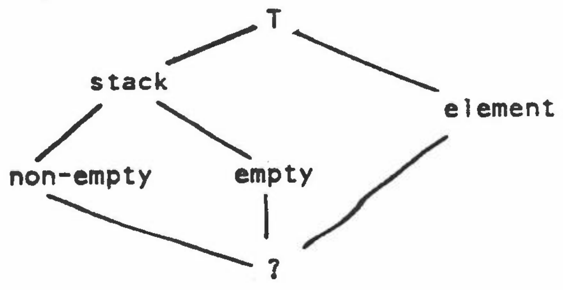
```
i)                           ii)
non-zero-real < real.        non-empty < stack.
zero < real.                 empty < stack.
                             ? < element.
```

The lattices corresponding to these type definitions are:-

i)



ii)



The reader is referred to [Cunningham and Dick 84] for a detailed treatment of lattice structured typing developed from related work by [Walther 83], [Shamir and Wadge 77] and others. One advantage of such typing is that it allows the treatment of some partial functions, which play an important role in many useful algebraic structures.

2.1.2. *Function signatures*

Function signatures are defined in the conventional way using the x and -> constructors on types; however, since the underlying type structure is a lattice, each function has a composite signature which is itself a lattice. Composite signatures are constructed from a set of templates which define how the function behaves on arguments of selected types. For example, consider function signatures for mult and inv which operate in the domain of the reals with the type lattice defined in Example 1 i). The templates shown in Example 2 are the smallest sets sufficient for the definition of mult and inv. Other templates (e.g. mult: real x real ->

real) may be included to make the definition more intuitive. Note that inv is a partial function which maps anything of type zero onto something of undefined type.

Example 2. Function signatures for mult and inv on the reals.

```
mult: real x non-zero-real -> real.
mult: non-zero-real x real -> real.
mult: non-zero-real x non-zero-real -> non-zero-real.
mult: zero x real -> zero.
mult: real x zero -> zero.

inv: non-zero -> non-zero.
inv: zero -> ?
```

A least-upper-bound interpretation of the template sets, which preserves the monotonicity of ->, is used to construct the composite function signatures. In this example the signatures include 16 elements for mult and 4 elements for inv:-

```
mult: real x real -> real.
mult: real x non-zero-real -> real.
mult: non-zero-real x real -> real.
mult: non-zero-real x non-zero-real -> non-zero-real.
mult: real x zero -> zero.
mult: zero x real -> zero.
mult: non-zero-real x zero -> zero.
mult: zero x non-zero-real -> zero.
mult: zero x zero -> zero.
mult: real x ? -> ?.
mult: ? x real -> ?.
mult: non-zero-real x ? -> ?.
mult: ? x non-zero-real -> ?.
mult: zero x ? -> ?.
mult: ? x zero -> ?.
mult: ? x ? -> ?.

inv: real -> non-zero-real.
inv: non-zero-real -> non-zero-real.
inv: zero -> ?.
inv: ? -> ?.
```

Constant symbols are treated as functions with no arguments, whose composite signatures contain just one element. For example, 0:zero defines the constant 0 to be of type zero.

2.1.3. *Variable name prefixes*

For convenience, ERIL allows a single character prefix to be associated with each type, to be used in the naming of variables. Thus instead of having to explicitly label variables by their type, e.g. x:real or y:non-zero-real, r and n can be declared, for example, as prefixes for real and non-zero-real variables respectively; any variable whose name begins with r is then known to be of type real, or with n of type non-zero-real.

2.2. EQUALITIES AND THEIR ATTRIBUTES

An equality is a (possibly ordered) pair of expressions in the algebra deemed to be equal in the equational theory. Associated with each equality is a unique identifier formed from a label (indicating to which set of equalities it currently belongs) and a number. In addition, information regarding the origin and derivation of both expressions is kept. The following subsections describe expressions, the equality predicate, and derivation history.

2.2.1. *Expressions*

Expressions are built in prefix notation from function symbols and variables in the conventional way; for example mult(n1,inv(n1)) and mult(0,x:real) are valid expressions in the context of Example 2. Variables may be labelled with types if necessary; unlabelled variables are assumed to follow the prefix conventions declared as part of the typing scheme. The type of an expression can be uniquely determined from the types of its primitive arguments according to the composite signatures of functions appearing in the expression.

2.2.2. *The equality predicate*

There are constraints on the types of expressions that can be regarded as being equal; these constraints correspond to a weak form of the equality predicate, which is undefined on arguments of undefined type, and false if the "meet" (or greatest lower bound) of the two argument types is the undefined type. Hence any expression of undefined type cannot form part of an equality pair, and the equality of two arguments whose "meet" type is undefined represents an inconsistency (the equality predicate is false). For example, inv(0) = 0 is not a valid equality, because inv(0) is of undefined type. The equality inv(x:non-zero-real) = 0 is actually false, since the type of inv(x:non-zero-real) is non-zero-real, and the meet of non-zero-real and zero is '?'.

2.2.3. *Derivation history*

The derivation history of each expression is recorded as a list of identifiers, maintained in strict order, of those equalities that have been applied to the expression upto its current form. The initial item in the list indicates where the expression originated, i.e. its original identifier and the name of the file from which it was read, or a reference to a critical expression and the equalities from which it was formed. The derivation history can be used to reconstruct proofs of hypotheses in terms of the initial axioms provided.

2.3. SETS OF EQUALITIES AND THEIR ATTRIBUTES

In an equational reasoning environment, it is often desirable to separate sets of equalities for documentational, algorithmic or other theoretical purposes; for example, two sets of rewrite-rules which are confluent within themselves. In addition, equalities may be grouped together to fulfill various roles. For example, some may be ordered for use as rewrite-rules; others may be unorderable, and thus only of use as undirected, two-way equations. For these reasons, ERIL allows the user to define any number of equality sets he chooses, identified by a unique label consisting of a single capital letter, and a name. Associated with each set are a number of attributes which describe the set itself, the ordering on expression pairs in equalities, how equalities within the set relate to one another, and how the set as a whole relates to other sets. The subsections which follow describe the set attributes.

2.3.1. *Kinds of equality set*

A set may be designated as containing one of (currently) four kinds of equality listed in Figure 1, where they are characterised by symbols representing the role that the equalities play.

Symbol	Characterisation
=	Unconsidered axioms (not yet formed into rules)
=>	Ordered rewrite-rules (uni-directional)
<=>	Equations (bi-directional rules)
=?=	Hypotheses (to be proven)

Figure 1. The four kinds of equality set

Amongst other things, the type attribute attaches to the set a particular algorithm (described in Section 3) for "applying" the equalities. It may be useful to define other kinds of equality with other forms of application, and ERIL is open to such extension at a future date.

2.3.2. *Ordering and sorting algorithms*

Two other algorithms may be associated with a set, one for ordering the pair of expressions in each equality, the other for sorting equalities within the set (e.g. into order of size). The ordering algorithm may succeed or fail, thus providing a test of qualification for membership in the set. Orderings are of course essential for rewrite-rules, and the ability to sort equalities within a set provides a mechanism for controlling the order in which rules are applied or selected. Any number of ordering and sorting algorithms could be invented, and ERIL is extendible in this regard. The default sorting on equalities if none other is specified is chronological order.

2.3.3. *Action lists*

Four (possibly empty) lists are associated with a set which describe how equalities in the set relate to each other and those in other sets. These lists are as follows:-

1) A list of labels of sets that are applied to this set. All equalities in the sets on the list are applied to those in the current set.

2) A list of labels of sets to which the current set is applied. The equalities in the current set are applied to all equalities in the sets on the list.

3) A list of labels of sets on which the left-hand side of equalities in the current set are to be superposed. Associated with each label is a flag with values "left", "right" or "both" indicating which side of the equality in the listed set is to be superposed.

4) A list of labels of sets on which the right-hand side of equalities in the current set are to be superposed. Labels are flagged in the same way as 3).

Again, the precise meaning of "applied" in the above depends on the kind of equalities involved as described in Section 3. It is possible for a set to be applied to itself or superposed on itself by including its own label in the appropriate action list.

3. *ERIL'S PROCEDURES*

ERIL's fundamental procedures are shown in Figure 3 in which arcs show dependencies. The following subsections describe various key procedures in more detail.

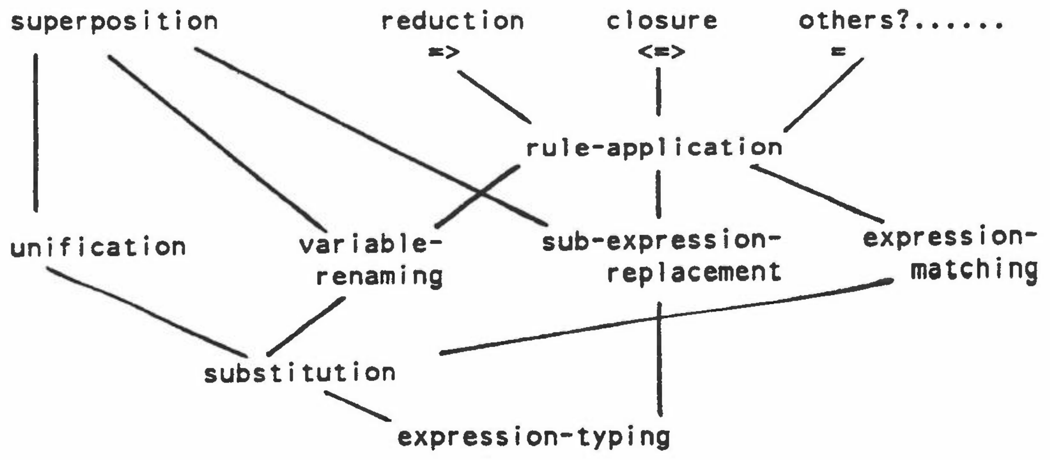


Figure 2. Dependency diagram for procedures in ERIL.

3.1. *Unification and Expression Matching*

It is known that unification on hierarchical types has a finite set of most general solutions [Walther 83], rather than the unique solution offered by unification in discretely typed first-order logic [Robinson 65]. ERIL's unification algorithm is an extension of that found in [Manna and Waldinger 81], specially adapted for lattice-structured types and the multiple solutions they entail. Details of these, as well as the expression-typing algorithm, may be found in [Cunningham and Dick 84].

3.2. *Reduction*

Reduction is the application of equalities of type => (rewrite rules), in other words the normalisation strategy. The two aspects are:-

- the order in which rules are applied to an expression to be reduced. This is controlled by the sorting algorithm placed on the set of rules, the top-most rule being applied first, and so forth.

- the order in which subexpressions are reduced by a rule. Currently ERIL adopts a strictly an outermost strategy.

3.3. *Closure*

Equalities of type <=> (equations) are applied to pairs of expressions to determine whether or not the two expressions are equivalent modulo the equations. A valid sequence of rule applications in this context is non-repeating, i.e. the expressions E and F are "closed" modulo a set of equations if there exists a sequence of rule applications

E <=> E' <=> E'' <=> ... <=> F

where none of E, E', ... E'' are identical. This ensures that the search space is finite. ERIL traverses the search space depth first.

3.4. *Other procedures*

Equalities of type = (unconsidered axioms) are applied simply by comparison for identity modulo renaming. Hypotheses (=?=) cannot be applied. ERIL is extensible in that other application procedures could be supplied and attached to new types of equality. Equally, new types of unification algorithm could be supplied, for instance an associative-commutative version, although no such experimentation has yet been done on ERIL.

4. OPERATION OF ERIL

All work is performed in ERIL when a new equality is introduced into a set, at which time the top level procedures hooked to the action lists on the set attributes are activated in strict order, as follows:-

i) the equality sets in action list 1) are applied to the new equality;

ii) the new equality is ordered by the ordering algorithm;

iii) the new equality is applied to the equality sets in list 2);

iv) the new equality is superposed on the sets in lists 3) and 4).

A typical completion strategy consists of moving equalities from an "unconsidered" set of type = into an initially empty set of type => which has left-hand superposition as an attribute. Critical pairs generated from superposition are replaced in the unconsidered set where they are reduced w.r.t. the => set. The operation stops when there are no more unconsidered equalities.

Commands in ERIL fall into three main categories: configuration commands, which allow algebras and set attributes to be established, movement commands, which actually cause the work to be done, and auxiliary commands for displaying sets and their derivation histories, writing sets to files, and other utilities.

The following three example applications will give a clear idea of the role these commands play in building completion/proof strategies. Actual session logs are given for these examples in the appendix, which are results from running a Waterloo Prolog version of ERIL on an IBM 4341 under VM/CMS.

4.1. *KNUTH-BENDIX COMPLETION OF A PARTIAL ALGEBRA*

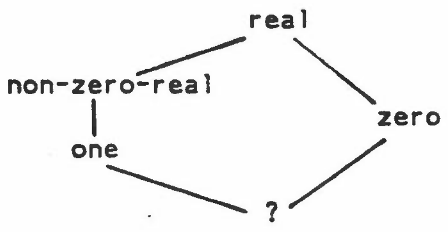
Two equality sets are used in this example, a set of rewrite-rules, R, and a set of unconsidered axioms, A, which have the following attributes:-

label:	R	
name:	Rewrite-rules	
type:	=>	
ordering:	kbord	(Knuth-Bendix style)
sorting:	bylhs	(smallest lhs sorted to the top)
action lists:	1) none	
	2) R A	(R reduces itself and A)
	3) R-left	(left-hand sides of R are superposed)
	4) none	(no right-hand sides are superposed)

label:	A	
name:	Unconsidered axioms	
type:	=	
ordering:	none	
sorting:	bysize	(smallest equality sorted to the top)
action lists:	1) R	(A is reduced w.r.t. R)
	2) none	
	3) none	
	4) none	

The example is based on the algebra definitions of mult and inv given in Section 2.1. The complete algebra definition is as follows:-

non-zero-real < real.
zero < real.
one < non-zero-real.



0: zero.
1: one.

```
mult: real x real -> real.           (Strictly speaking redundant)
mult: one x real -> real.
mult: real x one -> real.
mult: one x non-zero-real -> non-zero-real.
mult: non-zero-real x one -> non-zero-real.
mult: one x one -> one.
mult: one x zero -> zero.
mult: zero x one -> zero.
mult: zero x zero -> zero.

inv: real -> real.
inv: non-zero-real -> non-zero-real.
inv: zero -> ?.

x:real.                         (Variable prefixes)
r:non-zero-real.
```

The initial axioms provided to A are :-

```
A1   mult(0,x1) = 0
A2   mult(x1,0) = 0
A3   mult(1,x1) = x1
A4   mult(inv(x1),x1) = 1
A5   mult(mult(x1,x2),x3) = mult(x1,mult(x2,x3))
```

The algebra definitions are loaded from file or console, and the initial axioms to are loaded into set A. The entire completion process is now initiated by the following "move" command, shown here as it might appear during the session:-

```
m
ENTER label of equality to move: A
ENTER number of equality to move: all
ENTER label of destination set: R
ENTER label of set to collect new equalities: A
```

As each equality from A is moved, it is ordered and sorted into R, applied to all equalities in R and A, and superposed on equalities in R (including itself). Any rules in R that are reduced by the new rule, and critical pairs produced as a result of superposition, are fed back into A, where they are fully reduced w.r.t R and sorted into order of size. The move command finally terminates when there are no more equalities in A to be moved.

The sorting algorithm placed on A increases efficiency by causing the smallest axioms or critical pairs to be chosen before the larger ones. The sorting algorithm placed on R causes the smallest rules to be applied first, thus reducing the number of attempted matches.

When the move commands terminates, the following confluent set of 12 rewrite-rules remains in R:-

```
R11  inv(1) => 1
R1   mult(0,x1) => 0
R2   mult(x1,0) => 0
R3   mult(1,x1) => x1
R10  mult(x1,1) => x1
R12  inv(inv(x1)) => x1
R4   mult(inv(x1),x1) => 1
R13  mult(x1,inv(x1)) => 1
R17  inv(mult(x1,x2)) => mult(inv(x2),inv(x1))
R5   mult(mult(x1,x2),x3) => mult(x1,mult(x2,x3))
R6   mult(inv(x1),mult(x1,x2)) => x2
R14  mult(x1,mult(inv(x1),x2)) => x2
```

These rules are sorted as specified by the size of the left-hand size, and the missing numbers in the set represent intermediate rules that were reducible by later rules. In this example, the lattice-structured typing has allowed us to recognise the critical expression mult(inv(0),0) produced from superposing rules R2 and R4, as being invalid, because inv(0) is of undefined type. Such critical expressions are discarded; in this case, the critical pair, 1 = 0, that would have resulted speaks for itself!

4.2. PROOF OF A HYPOTHESIS

The next example shows how an equality set of kind =?= (hypotheses) behaves. Such equalities are not moved from their set when reduced by an equality from another set, but the lastest reduced form remains until proved (when the pair of expressions are identical).

The example chosen is an attempt to prove that a group in which all non-neutral elements are of order two is commutative. We are not required to produce a confluent set of rewrite-rules, but generate a set sufficient for the purposes of the proof. We will use a set of hypotheses, H, two sets of unconsidered axioms, A and B, and a set of rewrite-rules, R. The configurations are as follows, where A and B are identical:-

```
        label:  H
         name:  To be proved
         type:  =?=
     ordering:  none
      sorting:  none
 action lists:  1) R A B
                2) none
                3) none
                4) none
```

```
       label:  A
        name:  Unconsidered axioms
        type:  =
    ordering:  none
     sorting:  bysize         (smallest equality sorted to the top)
action lists:  1) R           (A is reduced w.r.t. R)
               2) H           (axioms are compared with H)
               3) none
               4) none
```

```
       label:  R
        name:  Rewrite-rules
        type:  =>
    ordering:  kbord
     sorting:  bylhs          (smallest lhs sorted to the top)
action lists:  1) none
               2) R H A B
               3) R-left
               4) none
```

The algebra definition has a single type, with functions f and e, as follows:-

```
? < s.

e: s.
f: s x s -> s.

x:s.
```

Given axioms:-

```
A1  f(e,x1) = x1
A2  f(x1,e) = x1
A3  f(x1,x1) = e
A4  f(f(x1,x2),x3) = f(x1,f(x2,x3))
```

Given hypothesis:-

```
H1  f(x1,x2) =?= f(x2,x1)
```

So as to maintain control of the proof process, we proceed in stages in which sets A and B alternate roles. First we move all of A into R, and collect new equalities in B; then we move all of B into R and collect new equalities in A, and so forth:-

```
m
ENTER label of equality to move: A
ENTER number of equality to move: all
ENTER label of destination set: R
ENTER label of set to collect new equalities: B
```

At the end of the first stage A is empty, and we have:-

```
R1    f(e,x1) => x1                        B5    x1 = f(x2,f(x2,x1))
R2    f(x1,e) => x1                        B4    f(x1,f(x2,f(x1,x2))) = e
R3    f(x1,x1) => e
R4    f(f(x1,x2),x3) => f(x1,f(x2,x3))     H1    f(x1,x2) =?= f(x2,x1)
```

```
m
ENTER label of equality to move: B
ENTER number of equality to move: all
ENTER label of destination set: R
ENTER label of set to collect new equalities: A
```

At the end of stage 2, B is empty, and we have:-

```
R1  f(e,x1) => x1                      A22 x1 = f(x2,f(x1,x2))
R2  f(x1,e) => x1                      A9  x1 = f(x2,f(x3,f(x2,f(x3,x1))))
R3  f(x1,x1) => e                      A11 f(x1,f(x2,f(x1,f(x2,x3)))) = x3
R4  f(f(x1,x2),x3) => f(x1,f(x2,x3))   A17 e = f(x1,f(x2,f(x3,f(x1,f(x2,x3)))))
R5  f(x1,f(x1,x2)) => x2
R6  f(x1,f(x2,f(x1,x2))) => e          H1  f(x1,x2) =?= f(x2,x1)
```

```
m
ENTER label of equality to move: A
ENTER number of equality to move: all
ENTER label of destination set: R
ENTER label of set to collect new equalities: B
```

During this stage, a new equality, B15, is produced which is found to be identical to H1, and H1 is marked as proved, at which point we have:-

```
R1  f(e,x1) => x1                      A9  x1 = f(x2,f(x3,f(x2,f(x3,x1))))
R2  f(x1,e) => x1                      A11 f(x1,f(x2,f(x1,f(x2,x3)))) = x3
R3  f(x1,x1) => e                      A17 e = f(x1,f(x2,f(x3,f(x1,f(x2,x3)))))
R4  f(f(x1,x2),x3) => f(x1,f(x2,x3))
R5  f(x1,f(x1,x2)) => x2               B15 f(x1,x2) = f(x2,x1)
R7  f(x1,f(x2,x1)) => x2               B13 f(x1,x2) = f(x3,f(x1,f(x3,x2)))
                                       B14 f(x1,f(x2,f(x3,x1))) = f(x2,x3)
H1  proved (identical to B15)          B12 x1 = f(x2,f(x3,f(x1,f(x2,x3))))
```

Using the "display derivation" facility, the following proof is given in terms of the initial axioms:-

```
f(x1,x2)  =  f(x1,f(x2,e))                             by A2
          =  f(x1,f(x2,f(f(x2,x1),f(x2,x1))))          by A3
          =  f(x1,f(x2,f(x2,f(x1,f(x2,x1)))))          by A4
          =  f(x1,f(f(x2,x2),f(x1,f(x2,x1))))          by A4
          =  f(x1,f(e,f(x1,f(x2,x1))))                 by A3
          =  f(x1,f(x1,f(x2,x1)))                      by A1
          =  f(f(x1,x1),f(x2,x1))                      by A4
          =  f(e,f(x2,x1))                             by A3
          =  f(x2,x1)                                  by A1
```

4.3. *COMPLETION MODULO A SET OF EQUATIONS*

Here we demonstrate the completion of a set of axioms with commutativity and associativity as equations. We use establish the following sets:-

```
        label:  A
         name:  Unconsidered axioms
         type:  =
     ordering:  none
      sorting:  bysize
 action lists:  1) R E     (A is reduced w.r.t. R and closed modulo E)
                2) none
                3) none
                4) none
```

```
        label:  R
         name:  Rewrite-rules
         type:  =>
     ordering:  kbord
      sorting:  none
 action lists:  1) none
                2) R E A
                3) R-left E-both    (Superposed on equations as well)
                4) none
        label:  E
         name:  Equations
         type:  <=>
     ordering:  none
      sorting:  none
 action lists:  1) none
                2) E A
                3) R-left        (Equations do not need to be
                4) R-left            superposed on themselves)
```

The single-sorted algebra definition is as follows:-

```
? < s.

0: s
1: s
E: s -> s
+: s x s -> s
.: s x s -> s

x:s
```

With initial axioms as follows:-

```
A1  E(x1 + x2) = E(x1).E(x2)        E1  (x1 + x2) + x3 = x1 + (x2 + x3)
A2  E(0) = 1                        E2  (x1.x2).x3 = x1.(x2.x3)
A3  0 + x1 = x1                     E3  x1 + x2 = x2 + x1
A4  1.x1 = x1                       E4  x1.x2 = x2.x1
```

The completion is performed with a single move operation:-

```
m
ENTER label of equality to move: A
ENTER number of equality to move: all
ENTER label of destination set: R
ENTER label of set to collect new equalities: A
```

When the move has finished, and A is empty, R contains the following 6 axioms:-

```
R1  E(x1 + x2) => E(x1).E(x2)
R2  E(0) => 1
R3  0 + x1 => x1
R4  1.x1 => x1
R5  x1 + 0 => x1
R6  x1.1 => x1
```

5. CONCLUSIONS

The architecture of ERIL seems to provide an interesting and useful environment for a wide variety of applications in equational reasoning. Its extensibility will allow room for much further experimentation. Future work will be directed to the following areas:-

- incorporating an AC-unification algorithm in order to build Peterson-Stickel type completion strategies [Peterson and Stickel 81].
- extending to a higher order algebra like KRC or ML.
- linking ERIL to a specification language, e.g. OBJ.
- improving the efficiency of the current Waterloo prolog implementation.
- gaining experience with using ERIL in a wide range of applications, particularly in program proving.

REFERENCES

[Baeten et al. 84]
J.C.M. Baeten, J.A. Bergstra, J.W. Klop
Priority Rewrite Systems
Report CS-R8407, Centre for Maths and Computer Science, P.O. Box 4079, 1009 AB Amsterdam

[Cunningham and Dick 84]
R.J. Cunningham, A.J.J. Dick
Rewrite Systems on a Lattice of Types
Report No. DOC 83/7, Imperial College, London SW7

[Forgaard and Guttag]
R. Forgaard, J.V. Guttag
REVE: A term Rewriting System Generator with Failure Resistant Knuth-Bendix
Proceedings of an NSF workshop on the rewrite rule laboratory, Sept 6-9 1983, General Electric CRD Rep. No. 84GEN008 pp. 33-56

[Huet 80]
G. Huet
Confluent Reductions: Abstract Properties and Applications to Term Rewrite Systems
JACM Vol. 27, No. 4 Oct 1980 pp. 797-821

[Huet and Hullot 82]
G. Huet, J-M Hullot
Proofs by Induction in Equational Theories with Constructors
JACM Vol. 25 No. 2 1982 pp. 239-266

[Kapur and Sivakumar 84]
D. Kapur, G. Sivakumar
Experiments with and Architecture of RRL, a Rewrite Rule Laboratory

Proceedings of an NSF workshop on the rewrite rule laboratory, Sept 6-9 1983, General Electric CRD Rep. No. 84GEN008 pp. 5-31

[Knuth and Bendix 70]
D. E. Knuth, P. B. Bendix
Simple Word Problems in Universal Algebras
In "Computational Problems and Abstract Algebras", J. Leech, Ed., Pergammon Press, 1970, pp. 263-297.

[Lescanne 83]
P. Lescanne
Computer Experiments with the REVE Term Rewriting System Generator
Proc. 10th Symp. on Principles of Programming Languages, ACM Austin, TX, Jan 1983 pp. 99-108

[Manna and Waldinger 81]
Z. Manna, R. Waldinger
Deductive Synthesis of the Unification Algorithm
Science of Computer Programming Vol. 1, 1981 pp. 5-48

[Peterson and Stickel 81]
G. E. Peterson, M. E. Stickel
Complete Sets of Reductions for Some Equational Theories
JACM Vol. 28, No. 2, April 1981, pp. 233-264

[Roberts]
G. Roberts
Waterloo Prolog Users' Manual
University of Waterloo, Ontario, Canada N2L 3G1

[Robinson 65]
J. A. Robinson
A Machine-Oriented Logic Based on the Resolution Principle
JACM Vol. 12, pp. 32-41

[Shamir and Wadge 77]
A. Shamir, W. W. Wadge
Data Types as Objects
LNCS 52 July 1977, pp. 465-479

[Walther 83]
C. Walther
A Many-Sorted Calculus Based on Resolution and Paramodulation
Proc. of the 8th International Joint Conf. on Artifical Intelligence, Karlsruhe 1983

[Walther 84]
C. Walther
Unification in Many-Sorted Theories
European Conference on Artificial Intelligence 1984

APPENDIX A

The console log of the completion of a partial algebra, corresponding to Section 4.1. The console output is a series of screens, initially showing the system configuration with each of the sets' attributes displayed in turn, followed by the MOVE command which performs the completion. The final screen shows some system statistics relevant to the run.

```
AVER Project - ERIL Prototype Tools         Imperial College - DoC    June 1984
------------------------Completion of a Partial Algebra------------------------
SYSTEM CONFIGURATION:                                      Console Log : on/OFF
E Q U A L I T Y   S E T S          Display      Sort     Order    Equality count
Label..Name..............Type........Trace....File.....File.....Total..Current
R  Rewrite-rules             =>       On   On     bylhs    kbord       0      0
A  Unconsidered axioms       =        On   Off    bysize   ____        5      5
--------------------------------------------------------------------------------
CONFIGURATION OPTIONS:                     Label  change equality set defaults
                                           n      create New equality set
                                           r      Reset configuration
                                           h      set screen Heading
                                           c      Console log on/off
                                           f      Finish configurations
--------------------------------------------------------------------------------
ENTER option:
R
```

```
AVER Project - ERIL Prototype Tools         Imperial College - DoC    June 1984
------------------------Completion of a Partial Algebra------------------------
Label ................ R                      | CONFIGURATION OPTIONS
Name ................. Rewrite-rules          | <- change       n
Type ................. =>                     | <- change       t
Auto-display/trace ... On /On                 | <- change       d
Sort algorithm ....... bylhs                  | <- change       s
Ordering algorithm ... kbord                  | <- change       o
Reduced by ........... R                      | <- change       r
Applied to ........... R A                    | <- change       a
LHS superposed with .. Rl                     | <- change       p
RHS superposed with ..                        | <- change       p
                                              |                 f    Finish
--------------------------------------------------------------------------------
ENTER option:
f A
```

```
AVER Project - ERIL Prototype Tools         Imperial College - DoC    June 1984
------------------------Completion of a Partial Algebra------------------------
Label ................ A                      | CONFIGURATION OPTIONS
Name ................. Unconsidered axioms    | <- change       n
Type ................. =                      | <- change       t
Auto-display/trace ... On /Off                | <- change       d
Sort algorithm ....... bysize                 | <- change       s
Ordering algorithm ... ____                   | <- change       o
Reduced by ........... R                      | <- change       r
Applied to ...........                        | <- change       a
LHS superposed with ..                        | <- change       p
RHS superposed with ..                        | <- change       p
                                              |                 f    Finish
--------------------------------------------------------------------------------
ENTER option:
f f
```

```
-----------------------Completion of a Partial Algebra-----------------------
R:    Rewrite-rules                   0
-----------------------------------------------------------------------------
A:    Unconsidered axioms             5
A1:   mult(0,x1)  =  0
A2:   mult(x1,0)  =  0
A3:   mult(1,x1)  =  x1
A4:   mult(inv(x1),x1)  =  1
A5:   mult(mult(x1,x2),x3)  =  mult(x1,mult(x2,x3))
-----------------------------------------------------------------------------
ENTER command:                                       (h for Help)
m
ENTER label of equality to move:                     (f to Finish, h for Help)
A all.
MOVING Aall:
ENTER label of destination set:                      (f to Finish)
R
ENTER label of set to receive new equalities:        (f to Finish)
A
MOVING A1:   mult(0,x1)  =  0   TO R.
Loading weight.....
R1:   mult(0,x1)  =>  0
Loading matched.....
Loading superpos.....
MOVING A2:   mult(x1,0)  =  0   TO R.
R2:   mult(x1,0)  =>  0
Loading replaced.....
MOVING A3:   mult(1,x1)  =  x1   TO R.
R3:   mult(1,x1)  =>  x1
MOVING A4:   mult(inv(x1),x1)  =  1   TO R.
R4:   mult(inv(x1),x1)  =>  1
MOVING A5:   mult(mult(x1,x2),x3)  =  mult(x1,mult(x2,x3))   TO R.
R5:   mult(mult(x1,x2),x3)  =>  mult(x1,mult(x2,x3))
MOVING A10:  x1  =  mult(inv(x2),mult(x2,x1))   TO R.
R6:   mult(inv(x1),mult(x1,x2))  =>  x2
MOVING A13:  mult(inv(1),x1)  =  x1   TO R.
R7:   mult(inv(1),x1)  =>  x1
MOVING A14:  mult(inv(inv(x1)),1)  =  x1   TO R.
R8:   mult(inv(inv(x1)),1)  =>  x1
MOVING A17:  mult(inv(inv(x1)),x2)  =  mult(x1,x2)   TO R.
R9:   mult(inv(inv(x1)),x2)  =>  mult(x1,x2)
REDUCING  R8:   mult(x1,1)  =>  x1
MOVING A23:  mult(x1,1)  =  x1   TO R.
R10:  mult(x1,1)  =>  x1
MOVING A29:  inv(1)  =  1   TO R.
R11:  inv(1)  =>  1
REDUCING  R7:   mult(1,x1)  =>  x1
MOVING A33:  inv(inv(x1))  =  x1   TO R.
R12:  inv(inv(x1))  =>  x1
REDUCING  R9:   mult(x1,x2)  =>  mult(x1,x2)
MOVING A25:  mult(x1,inv(x1))  =  1   TO R.
R13:  mult(x1,inv(x1))  =>  1
MOVING A27:  mult(x1,mult(inv(x1),x2))  =  x2   TO R.
R14:  mult(x1,mult(inv(x1),x2))  =>  x2
MOVING A46:  1  =  mult(x1,mult(x2,inv(mult(x1,x2))))   TO R.
R15:  mult(x1,mult(x2,inv(mult(x1,x2))))  =>  1
MOVING A70:  inv(x1)  =  mult(x2,inv(mult(x1,x2)))   TO R.
R16:  mult(x1,inv(mult(x2,x1)))  =>  inv(x2)
REDUCING  R15:  mult(x1,inv(x1))  =>  1
MOVING A85:  mult(inv(x1),inv(x2))  =  inv(mult(x2,x1))   TO R.
R17:  inv(mult(x1,x2))  =>  mult(inv(x2),inv(x1))
REDUCING  R16:  mult(x1,mult(inv(x1),inv(x2)))  =>  inv(x2)
MOVE FINISHED.
```

```
AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
------------------------Completion of a Partial Algebra------------------------
R:    Rewrite-rules                 12
R11:  inv(1)  =>  1
R1:   mult(0,x1)  =>  0
R2:   mult(x1,0)  =>  0
R3:   mult(1,x1)  =>  x1
R10:  mult(x1,1)  =>  x1
R12:  inv(inv(x1))  =>  x1
R4:   mult(inv(x1),x1)  =>  1
R13:  mult(x1,inv(x1))  =>  1
R17:  inv(mult(x1,x2))  =>  mult(inv(x2),inv(x1))
R5:   mult(mult(x1,x2),x3)  =>  mult(x1,mult(x2,x3))
R6:   mult(inv(x1),mult(x1,x2))  =>  x2
R14:  mult(x1,mult(inv(x1),x2))  =>  x2
-------------------------------------------------------------------------------
A:    Unconsidered axioms           0
-------------------------------------------------------------------------------
ENTER command:                                        (h for Help)
d
Loading display.....


AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
------------------------Completion of a Partial Algebra------------------------
DISPLAY OPTIONS:       Label  display equality set
                       a      display Algebra definitions
                       l      display domain Lattice
                       d      display Derivation
                       s      display system Statistics
                       c      display system Configuration
                       f      Finish display option
-------------------------------------------------------------------------------
ENTER option:
s


AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
------------------------Completion of a Partial Algebra------------------------
NUMBER OF MATCHES        Attempts   : 1859
                         Successful: 186

NUMBER OF UNIFICATIONS   Attempts   : 178
                         Successful: 114

NUMBER OF CRITICAL EXPRESSIONS      : 92
-------------------------------------------------------------------------------
DISPLAY OPTIONS:       Label  display equality set
                       a      display Algebra definitions
                       l      display domain Lattice
                       d      display Derivation
                       s      display system Statistics
                       c      display system Configuration
                       f      Finish display option
-------------------------------------------------------------------------------
ENTER option:
f b
```

```
AVER Project - ERIL Prototype Tools        Imperial College - DoC    June 1984
------------------------Completion of a Partial Algebra------------------------
SYSTEM CONFIGURATION:                                     Console Log : on/OFF
E Q U A L I T Y   S E T S       Display       Sort     Order    Equality count
Label..Name...............Type........Trace....File.....File.....Total..Current
R  Rewrite-rules           =>      On    On      bylhs    kbord       17     12
A  Unconsidered axioms     =       On    Off     bysize   ____       102      0
-------------------------------------------------------------------------------
CONFIGURATION OPTIONS:                      Label  change equality set defaults
                                            n      create New equality set
                                            r      Reset configuration
                                            h      set screen Heading
                                            c      Console log on/off
                                            f      Finish configurations
-------------------------------------------------------------------------------
ENTER option:
f f
CONFIRM exit from ERIL (y/n):
y
```

APPENDIX B

The console log of the proof of commutativity of the group in which all non-neutral elements are of order two. corresponding to Section 4.2. The console output is a series of screens, initially showing the system configuration and the attributes of the four equality sets involved, followed the three MOVE commands which perform the proof.

```
AVER Project - ERIL Prototype Tools         Imperial College - DoC    June 1984
-----------------------Proof that a group is commutative----------------------
SYSTEM CONFIGURATION:                                       Console Log : on/OFF
E Q U A L I T Y   S E T S        Display       Sort     Order    Equality count
Label..Name..............Type........Trace....File.....File.....Total..Current
A  Unconsidered axioms    =      On    Off     bysize    ____        4      4
B  Unconsidered axioms    =      On    Off     bysize    ____        0      0
R  Rewrite-rules          =>     On    On      bylhs     kbord       0      0
H  Hypotheses             =?=    On    On      ____      ____        1      1
-------------------------------------------------------------------------------
CONFIGURATION OPTIONS:                   Label   change equality set defaults
                                         n       create New equality set
                                         r       Reset configuration
                                         h       set screen Heading
                                         c       Console log on/off
                                         f       Finish configurations
-------------------------------------------------------------------------------
ENTER option:
A
```

```
AVER Project - ERIL Prototype Tools         Imperial College - DoC    June 1984
-----------------------Proof that a group is commutative----------------------
Label ................ A                     | CONFIGURATION OPTIONS
Name ................. Unconsidered axioms   | <- change        n
Type ................. =                     | <- change        t
Auto-display/trace ... On /Off               | <- change        d
Sort algorithm ....... bysize                | <- change        s
Ordering algorithm ... ____                  | <- change        o
Reduced by ........... R                     | <- change        r
Applied to ........... H                     | <- change        a
LHS superposed with ..                       | <- change        p
RHS superposed with ..                       | <- change        p
                                             |                  f    Finish
-------------------------------------------------------------------------------
ENTER option:
f B
```

```
AVER Project - ERIL Prototype Tools        Imperial College - DoC    June 1984
-----------------------Proof that a group is commutative----------------------
Label ............... B                     |  CONFIGURATION OPTIONS
Name ................ Unconsidered axioms   |  <- change        n
Type ................ =                     |  <- change        t
Auto-display/trace ... On /Off              |  <- change        d
Sort algorithm ....... bysize               |  <- change        s
Ordering algorithm ... ____                 |  <- change        o
Reduced by .......... R                     |  <- change        r
Applied to .......... H                     |  <- change        a
LHS superposed with ..                      |  <- change        p
RHS superposed with ..                      |  <- change        p
                                            |                   f    Finish
------------------------------------------------------------------------------
ENTER option:
f R
```

```
AVER Project - ERIL Prototype Tools        Imperial College - DoC    June 1984
-----------------------Proof that a group is commutative----------------------
Label ............... R                     |  CONFIGURATION OPTIONS
Name ................ Rewrite-rules         |  <- change        n
Type ................ =>                    |  <- change        t
Auto-display/trace ... On /On               |  <- change        d
Sort algorithm ....... bylhs                |  <- change        s
Ordering algorithm ... kbord                |  <- change        o
Reduced by .......... R                     |  <- change        r
Applied to .......... R H A B               |  <- change        a
LHS superposed with .. Rl                   |  <- change        p
RHS superposed with ..                      |  <- change        p
                                            |                   f    Finish
------------------------------------------------------------------------------
ENTER option:
f H
```

```
AVER Project - ERIL Prototype Tools        Imperial College - DoC    June 1984
-----------------------Proof that a group is commutative----------------------
Label ............... H                     |  CONFIGURATION OPTIONS
Name ................ Hypotheses            |  <- change        n
Type ................ =?=                   |  <- change        t
Auto-display/trace ... On /On               |  <- change        d
Sort algorithm ....... ____                 |  <- change        s
Ordering algorithm ... ____                 |  <- change        o
Reduced by .......... R A B                 |  <- change        r
Applied to ..........                       |  <- change        a
LHS superposed with ..                      |  <- change        p
RHS superposed with ..                      |  <- change        p
                                            |                   f    Finish
------------------------------------------------------------------------------
ENTER option:
f f
```

```
----------------------Proof that a group is commutative----------------------
A:    Unconsidered axioms           4
A1:   f(e,x1)  =  x1
A2:   f(x1,e)  =  x1
A3:   f(x1,x1)  =  e
A4:   f(f(x1,x2),x3)  =  f(x1,f(x2,x3))
-----------------------------------------------------------------------------
B:    Unconsidered axioms           0
-----------------------------------------------------------------------------
R:    Rewrite-rules                 0
-----------------------------------------------------------------------------
H:    Hypotheses                    1
H1:   f(x1,x2)  =?=  f(x2,x1)
    ├ f(x1,x2)  =?=  f(x2,x1) ┤
-----------------------------------------------------------------------------
ENTER command:                                  (h for Help)
m
ENTER label of equality to move:                (f to Finish, h for Help)
A all.
MOVING Aall:
ENTER label of destination set:                 (f to Finish)
R
ENTER label of set to receive new equalities:   (f to Finish)
B
MOVING A1:   f(e,x1)  =  x1   TO R.
Loading weight.....
R1:   f(e,x1)  =>  x1
Loading matched.....
Loading superpos.....
MOVING A2:   f(x1,e)  =  x1   TO R.
R2:   f(x1,e)  =>  x1
Loading replaced.....
MOVING A3:   f(x1,x1)  =  e   TO R.
R3:   f(x1,x1)  =>  e
MOVING A4:   f(f(x1,x2),x3)  =  f(x1,f(x2,x3))   TO R.
R4:   f(f(x1,x2),x3)  =>  f(x1,f(x2,x3))
MOVE FINISHED.
```

```
-----------------------Proof that a group is commutative----------------------
A:    Unconsidered axioms           0
------------------------------------------------------------------------------
B:    Unconsidered axioms           2
B5:   x1  =  f(x2,f(x2,x1))
B4:   f(x1,f(x2,f(x1,x2)))  =  e
------------------------------------------------------------------------------
R:    Rewrite-rules                 4
R1:   f(e,x1)  =>  x1
R2:   f(x1,e)  =>  x1
R3:   f(x1,x1)  =>  e
R4:   f(f(x1,x2),x3)  =>  f(x1,f(x2,x3))
------------------------------------------------------------------------------
H:    Hypotheses                    1
H1:   f(x1,x2)  =?=  f(x2,x1)
    ├ f(x1,x2)  =?=  f(x2,x1) ┤
------------------------------------------------------------------------------
ENTER command:                                   (h for Help)
m
ENTER label of equality to move:                 (f to Finish, h for Help)
B all.
MOVING Ball:
ENTER label of destination set:                  (f to Finish)
R
ENTER label of set to receive new equalities:    (f to Finish)
A
MOVING B5:   x1  =  f(x2,f(x2,x1))   TO R.
R5:   f(x1,f(x1,x2))  =>  x2
MOVING B4:   f(x1,f(x2,f(x1,x2)))  =  e   TO R.
R6:   f(x1,f(x2,f(x1,x2)))  =>  e
MOVE FINISHED.
```

```
-----------------------Proof that a group is commutative----------------------
A:    Unconsidered axioms          4
A22:  x1  =  f(x2,f(x1,x2))
A9:   x1  =  f(x2,f(x3,f(x2,f(x3,x1))))
A11:  f(x1,f(x2,f(x1,f(x2,x3))))  =  x3
A17:  e  =  f(x1,f(x2,f(x3,f(x1,f(x2,x3)))))
------------------------------------------------------------------------------
B:    Unconsidered axioms          0
------------------------------------------------------------------------------
R:    Rewrite-rules                6
R1:   f(e,x1)  =>  x1
R2:   f(x1,e)  =>  x1
R3:   f(x1,x1)  =>  e
R4:   f(f(x1,x2),x3)  =>  f(x1,f(x2,x3))
R5:   f(x1,f(x1,x2))  =>  x2
R6:   f(x1,f(x2,f(x1,x2)))  =>  e
------------------------------------------------------------------------------
H:    Hypotheses                   1
H1:   f(x1,x2)  =?=  f(x2,x1)
    ⊢ f(x1,x2)  =?=  f(x2,x1) ⊣
------------------------------------------------------------------------------
ENTER command:                                   (h for Help)
m
ENTER label of equality to move:                 (f to Finish, h for Help)
A all.
MOVING Aall:
ENTER label of destination set:                  (f to Finish)
R
ENTER label of set to receive new equalities:    (f to Finish)
B
MOVING A22:  x1  =  f(x2,f(x1,x2))   TO R.
R7:   f(x1,f(x2,x1))  =>  x2
REDUCING  R6:   f(x1,x1)  =>  e
PROVED    H1:   f(x1,x2)  =?=  f(x1,x2)
```

```
-----------------------Proof that a group is commutative----------------------
PAUSE:

PROVED    H1:   f(x1,x2)  =?=  f(x1,x2)


ENTER Continue or Halt  (c/h) :
h
MOVE FINISHED.
```

```
AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
-----------------------Proof that a group is commutative-----------------------
A:     Unconsidered axioms            3
A9:    x1  =  f(x2,f(x3,f(x2,f(x3,x1))))
A11:   f(x1,f(x2,f(x1,f(x2,x3))))  =  x3
A17:   e  =  f(x1,f(x2,f(x3,f(x1,f(x2,x3)))))
--------------------------------------------------------------------------------
B:     Unconsidered axioms            4
B15:   f(x1,x2)  =  f(x2,x1)
B13:   f(x1,x2)  =  f(x3,f(x1,f(x3,x2)))
B14:   f(x1,f(x2,f(x3,x1)))  =  f(x2,x3)
B12:   x1  =  f(x2,f(x3,f(x1,f(x2,x3))))
--------------------------------------------------------------------------------
R:     Rewrite-rules                  6
R1:    f(e,x1)  =>  x1
R2:    f(x1,e)  =>  x1
R3:    f(x1,x1)  =>  e
R4:    f(f(x1,x2),x3)  =>  f(x1,f(x2,x3))
R5:    f(x1,f(x1,x2))  =>  x2
R7:    f(x1,f(x2,x1))  =>  x2
--------------------------------------------------------------------------------
H:     Hypotheses                     1
H1:    f(x1,x2)  =?=  f(x2,x1)
     ⊦ PROVED ⊣
--------------------------------------------------------------------------------
ENTER command:                                    (h for Help)
d
Loading display.....
```

```
AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
-----------------------Proof that a group is commutative-----------------------
DISPLAY OPTIONS:       Label  display equality set
                       a      display Algebra definitions
                       l      display domain Lattice
                       d      display Derivation
                       s      display system Statistics
                       c      display system Configuration
                       f      Finish display option
--------------------------------------------------------------------------------
ENTER option:
s
```

```
AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
-----------------------Proof that a group is commutative----------------------
NUMBER OF MATCHES          Attempts  : 424
                           Successful: 48

NUMBER OF UNIFICATIONS     Attempts  : 52
                           Successful: 42

NUMBER OF CRITICAL EXPRESSIONS       : 37
-------------------------------------------------------------------------------
DISPLAY OPTIONS:         Label  display equality set
                         a      display Algebra definitions
                         l      display domain Lattice
                         d      display Derivation
                         s      display system Statistics
                         c      display system Configuration
                         f      Finish display option
-------------------------------------------------------------------------------
ENTER option:
f b
```

```
AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
-----------------------Proof that a group is commutative----------------------
SYSTEM CONFIGURATION:                                       Console Log : on/OFF
E Q U A L I T Y   S E T S         Display        Sort      Order    Equality count
Label..Name...............Type........Trace....File.....File.....Total..Current
A  Unconsidered axioms     =          On   Off    bysize    ____       25      3
B  Unconsidered axioms     =          On   Off    bysize    ____       17      4
R  Rewrite-rules           =>         On   On     bylhs     kbord       7      6
H  Hypotheses              =?=        On   On     ____      ____        1      1
-------------------------------------------------------------------------------
CONFIGURATION OPTIONS:                      Label  change equality set defaults
                                            n      create New equality set
                                            r      Reset configuration
                                            h      set screen Heading
                                            c      Console log on/off
                                            f      Finish configurations
-------------------------------------------------------------------------------
ENTER option:
f f
CONFIRM exit from ERIL (y/n):
y
```

APPENDIX C

The console log of a completion modulo a set of equations, corresponding to Section 4.3.

```
AVER Project - ERIL Prototype Tools         Imperial College - DoC    June 1984
--------------------------Completion modulo Equations-------------------------
SYSTEM CONFIGURATION:                                      Console Log : on/OFF
E Q U A L I T Y   S E T S          Display       Sort     Order    Equality count
Label..Name...............Type........Trace....File.....File.....Total..Current
A  Unconsidered axioms     =         On   Off     bysize    ____        4      4
R  Rewrite-rules           =>        On   On      bylhs     kbord       0      0
E  Equations               <=>       On   On      bysize    ____        4      4
-------------------------------------------------------------------------------
CONFIGURATION OPTIONS:                     Label  change equality set defaults
                                           n      create New equality set
                                           r      Reset configuration
                                           h      set screen Heading
                                           c      Console log on/off
                                           f      Finish configurations
-------------------------------------------------------------------------------
ENTER option:
A
```

```
AVER Project - ERIL Prototype Tools         Imperial College - DoC    June 1984
-------------------------Completion modulo Equations--------------------------
Label ................ A                    |  CONFIGURATION OPTIONS
Name ................. Unconsidered axioms  |  <- change       n
Type ................. =                    |  <- change       t
Auto-display/trace ... On /Off              |  <- change       d
Sort algorithm ....... bysize               |  <- change       s
Ordering algorithm ... ____                 |  <- change       o
Reduced by ........... R E                  |  <- change       r
Applied to ..........                       |  <- change       a
LHS superposed with ..                      |  <- change       p
RHS superposed with ..                      |  <- change       p
                                            |                  f    Finish
-------------------------------------------------------------------------------
ENTER option:
f R
```

```
AVER Project - ERIL Prototype Tools        Imperial College - DoC    June 1984
-------------------------Completion modulo Equations-------------------------
Label ................ R                     | CONFIGURATION OPTIONS
Name ................. Rewrite-rules         | <- change         n
Type ................. =>                    | <- change         t
Auto-display/trace ... On /On                | <- change         d
Sort algorithm ....... bylhs                 | <- change         s
Ordering algorithm ... kbord                 | <- change         o
Reduced by ........... R E                   | <- change         r
Applied to ........... R E A                 | <- change         a
LHS superposed with .. Rl El                 | <- change         p
RHS superposed with ..                       | <- change         p
                                             |                   f    Finish
-----------------------------------------------------------------------------
ENTER option:
f E
```

```
AVER Project - ERIL Prototype Tools        Imperial College - DoC    June 1984
-------------------------Completion modulo Equations-------------------------
Label ................ E                     | CONFIGURATION OPTIONS
Name ................. Equations             | <- change         n
Type ................. <=>                   | <- change         t
Auto-display/trace ... On /On                | <- change         d
Sort algorithm ....... bysize                | <- change         s
Ordering algorithm ... ____                  | <- change         o
Reduced by ........... R                     | <- change         r
Applied to ........... R E A                 | <- change         a
LHS superposed with .. Rl                    | <- change         p
RHS superposed with ..                       | <- change         p
                                             |                   f    Finish
-----------------------------------------------------------------------------
ENTER option:
f f
```

```
--------------------------Completion modulo Equations-------------------------
A:    Unconsidered axioms            4
A1:   i(0)  =  1
A2:   (s1+0)  =  s1
A3:   s1.1  =  s1
A4:   i(s1+s2)  =  i(s1).i(s2)
------------------------------------------------------------------------------
R:    Rewrite-rules                  0
------------------------------------------------------------------------------
E:    Equations                      4
E1:   (s1+s2)  <=>  (s2+s1)
E2:   (s1.s2)  <=>  (s2.s1)
E3:   (s1+s2+s3)  <=>  ((s1+s2)+s3)
E4:   (s1.s2.s3)  <=>  ((s1.s2).s3)
------------------------------------------------------------------------------
ENTER command:                                 (h for Help)
m
ENTER label of equality to move:               (f to Finish, h for Help)
A all.
MOVING Aall:
ENTER label of destination set:                (f to Finish)
R
ENTER label of set to receive new equalities:  (f to Finish)
A
MOVING A1:   i(0)  =  1   TO R.
Loading weight.....
R1:   i(0)  =>  1
Loading superpos.....
MOVING A2:   (s1+0)  =  s1   TO R.
R2:   (s1+0)  =>  s1
MOVING A3:   s1.1  =  s1   TO R.
R3:   (s1.1)  =>  s1
MOVING A5:   s1  =  (0+s1)   TO R.
R4:   (0+s1)  =>  s1
MOVING A7:   s1  =  1.s1   TO R.
R5:   (1.s1)  =>  s1
MOVING A4:   i(s1+s2)  =  i(s1).i(s2)   TO R.
R6:   i(s1+s2)  =>  (i(s1).i(s2))
MOVE FINISHED.
```

```
AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
--------------------------Completion modulo Equations--------------------------
A:    Unconsidered axioms            0
-------------------------------------------------------------------------------
R:    Rewrite-rules                  6
R1:   i(0)  =>  1
R2:   (s1+0)  =>  s1
R3:   (s1.1)  =>  s1
R4:   (0+s1)  =>  s1
R5:   (1.s1)  =>  s1
R6:   i(s1+s2)  =>  (i(s1).i(s2))
-------------------------------------------------------------------------------
E:    Equations                      4
E1:   (s1+s2)  <=>  (s2+s1)
E2:   (s1.s2)  <=>  (s2.s1)
E3:   (s1+s2+s3)  <=>  ((s1+s2)+s3)
E4:   (s1.s2.s3)  <=>  ((s1.s2).s3)
-------------------------------------------------------------------------------
ENTER command:                                    (h for Help)
d
Loading display.....
```

```
AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
--------------------------Completion modulo Equations--------------------------
DISPLAY OPTIONS:       Label  display equality set
                       a      display Algebra definitions
                       l      display domain Lattice
                       d      display Derivation
                       s      display system Statistics
                       c      display system Configuration
                       f      Finish display option
-------------------------------------------------------------------------------
ENTER option:
s
```

```
AVER Project - ERIL Prototype Tools          Imperial College - DoC    June 1984
--------------------------Completion modulo Equations--------------------------
NUMBER OF MATCHES         Attempts  : 213
                          Successful: 51

NUMBER OF UNIFICATIONS    Attempts  : 19
                          Successful: 16

NUMBER OF CRITICAL EXPRESSIONS      : 14
-------------------------------------------------------------------------------
DISPLAY OPTIONS:       Label  display equality set
                       a      display Algebra definitions
                       l      display domain Lattice
                       d      display Derivation
                       s      display system Statistics
                       c      display system Configuration
                       f      Finish display option
-------------------------------------------------------------------------------
ENTER option:
f b
```

```
AVER Project - ERIL Prototype Tools        Imperial College - DoC    June 1984
--------------------------Completion modulo Equations-------------------------
SYSTEM CONFIGURATION:                                       Console Log : on/OFF
E Q U A L I T Y   S E T S        Display       Sort    Order    Equality count
Label..Name...............Type.......Trace....File.....File.....Total..Current
A  Unconsidered axioms    =        On   Off    bysize   ____       18      0
R  Rewrite-rules          =>       On   On     bylhs    kbord       6      6
E  Equations              <=>      On   On     bysize   ____        4      4
-------------------------------------------------------------------------------
CONFIGURATION OPTIONS:                      Label  change equality set defaults
                                            n      create New equality set
                                            r      Reset configuration
                                            h      set screen Heading
                                            c      Console log on/off
                                            f      Finish configurations
-------------------------------------------------------------------------------
ENTER option:
f f
CONFIRM exit from ERIL (y/n):
y
```