RAL-92-025

Science and Engineering Research Council
Rutherford Appleton Laboratory
Chilton DIDCOT Oxon OX11 0QX

RAL-92-025

Experiments with Strategies for Equational Reasoning

B M Matthews

May 1992

Experiments with Strategies for Equational Reasoning.

Brian M. Matthews.[1]

Systems Engineering Division,
Informatics Department,
Rutherford Appleton Laboratory,
Didcot,
OXON, OX11 0QX

April 9, 1992

ABSTRACT

Equational Reasoning is a powerful method for carrying out formal reasoning within an automated environment. However, typical equational reasoning inferences are sensitive to the strategy in which they are applied as to their efficiency. In this paper we present a series of different strategies which can be used for a variety of equational reasoning problems. The ERIL equational reasoning system has been designed to be highly configurable to allow flexibility in strategy. We describe experiments employing configurations of the ERIL system for the presented strategies for Knuth-Bendix Completion, Theorem Proving in Horn Clause Logic using an equational method, and Unfailing Completion.

1. Introduction

1.1. Equational Reasoning

Equational Reasoning is the process of carrying out mathematical or logical reasoning by replacing equals by equals. This form of reasoning is to be found throughout mathematics and science as it natural to express laws and relations between entities by what constitutes identity in the domain of discourse. This use of equations to formally express properties extends to computing science where equational reasoning is an important technique in the fields of, for example, automated theorem proving, program transformation, symbolic execution and programming, and program specification and verification.

The laws of equational reasoning give a powerful method of reasoning using a (finite) set of equations, known as axioms, to capture a theory. However, the problem with the

[1] Work supported by the SERC EASE+ programme and the IEATP project IED/1477 "Verification Techniques for LOTOS". The author now at Dept. of Computing Science, University of Glasgow, 17 Lilybank Gardens, Glasgow, G12 8QQ.

uncontrolled use of this is the very large search space involved to find a solution. To control this, the equations can be oriented into *rewrite rules*, so that replacements of equals by equals can only be made in one direction: once a rewrite rule has been applied to a subterm, it cannot be applied in the opposite direction to reverse the inference step. Once this has been done these rules can be applied automatically. However, the problem then arises that a set of axioms converted into a set of rewrite rules loses some of their reasoning power. Equals cannot be replaced by equals if it requires using a rewrite rule in reverse, and thus we cannot prove the same set of theorems in the equational theory as held previously. We have lost *Completeness*. We would like to convert a set of axioms into a set of rewrite rules and maintain the reasoning power of the original equational theory. The Knuth-Bendix completion procedure attempts to regain the lost power.

The Knuth-Bendix procedure can be adapted to carry out further theorem proving tasks. If rules for horn-clause logic are formulated equationally, the procedure can be used to carry out refutational theorem proving, by attempting to generate a contradiction through equational inferences. Also, although the Knuth-Bendix procedure may fail to terminate with a decision procedure, by a process similar to Skolemisation a refutational semi-decision procedure known as Unfailing Completion can be used.

All the equational reasoning techniques can be expressed as a series of inference rules. However, the efficiency and effectiveness of the application of these procedures is sensitive to the order in which they are applied. This order of application or strategy is investigated further in this paper.

For more general introduction to the ideas of equational reasoning see [Dick91] whilst for a formal treatment see [HuOp80, DerJou90]. Several theorem provers have been produced which use implement a variety of equational techniques. These include REVE [Forg84], the Larch Prover [GarGut89], and RRL [KapZha89a]. In this paper, we describe the implementation of these strategies in terms of the ERIL system [Dick87, DicKal88].

1.2. Notations and Definitions.

We give a brief overview of the basic set of notational conventions and definitions used in this paper.

Given a set of *operator* symbols Σ, such that each $f \in \Sigma$ has an *arity*, denoted, $\alpha(f)$ which is a fixed natural number, and a countable set of *variable* symbols X such that $X \cap \Sigma = \emptyset$, we define the set of terms $T(\Sigma,X)$ to be the either a member of X or a string of the form $f(t_1, \cdots, t_{\alpha(f)})$ where $f \in \Sigma$ and $t_1, \cdots, t_{\alpha(f)} \in T(\Sigma,X)$. The variable free terms (*ground* terms) are denoted by $T(\Sigma)$. An algebra for $T(\Sigma)$ is any set A, known as the *carrier* of Σ, a function $D^A : T(\Sigma) \rightarrow A$ together with a set of functions F such that for every operator $f \in \Sigma$ there is a $f^A : (A^{\alpha(f)} \rightarrow A) \in F$, of arity $\alpha(f)$, such that $D^A(f(t_1, \cdots, t_{\alpha(f)})) = f^A(D^A(t_1), \cdots, D^A(t_{\alpha(f)}))$. In particular the set of terms is an algebra for itself (the *Term Algebra*).

Definition. A *substitution*, denoted σ is an endomorphism on $T(\Sigma,X)$. The domain of a substitution is a set of variables $Dom(\sigma)$ such that $\sigma(x)=x$ if $x \notin Dom(\sigma)$.

Definition. An *occurrence* is a possibly empty sequence of natural numbers, written Λ for the empty string and $n.s$ for the prefix n on the string s. The set of occurrences of a term $M(t)$ is defined to be $M(f(t_1, \cdots, t_{\alpha(f)})) = \{\Lambda\} \cup \{i.n \mid i \in \{1 \cdots \alpha(f)\} \wedge n \in M(t_i)\}$.

If $t \in T$ is a term and u an occurrence in $M(t)$, then we denote by t/u the subterm occurring at u, given by $t/\Lambda = t$ and $f(t_1, \cdots, t_{\alpha(f)})/i.u = t_i/u$. Also we denote by $t[u \leftarrow s]$ the term resulting from replacing the term t/u in t by the term s, given that $u \in M(t)$. So $t[u \leftarrow s]/u = s$. In this paper, it is assumed that whenever t/u is used, the occurrence u in t is well-defined

Definition. An *Equation* is a pair of terms denoted $s = t$. An algebra A satisfies an equation $s = t$ denoted $A \models s = t$ iff $\forall \sigma$, $D^A(\sigma(s)) = D^A(\sigma(t))$.

This notion is formally captured by the four laws of equational logic.

Definition. A *congruence relation* = on terms, obeys the following rules of equational logic.

1 **Reflexivity:** $\forall x \in T.\ x = x$.

2 **Symmetry:** $\forall x,y \in T.\ x = y \Rightarrow y = x$.

3 **Transitivity:** $\forall x,y,z \in T.\ x = y \wedge y = z \Rightarrow x = z$.

4 **Congruence:** $\forall x,y,t \in T.\ x = y \Rightarrow t[u \leftarrow x] = t[u \leftarrow y]$ for any occurrence u in t.

We give theories in equational logic by a finite set of equations as described in the following definition.

Definition. An *Equational Theory* over a term algebra $T(\Sigma,X)$ is the set of congruence classes induced by a set of equations $E = \{s_i = t_i\}_{i=1}^n$, where $\forall i\ .\ s_i, t_i \in T(\Sigma,X)$. If two terms s and t are found to be be in the same congruence class using the axioms E, under the rules of equational logic, we denote this by $s =_E t$.

Definition. A *Rewrite Rule* is an ordered pair of terms from $T(\Sigma,X)$, written $l \rightarrow r$.

Definition. A *Term-Rewriting System* is a finite set of rewrite rules $\{l_i \rightarrow r_i\}_{i=1}^n$ such that all variables in r_i occur in l_i.

Before we define the rewrite relation on a term algebra, there are two definitions which formalise the properties such a relation should have.

Definition: A relation $\sim \subset T(\Sigma,X) \times T(\Sigma,X)$ is *Stable* if $\forall$ substitutions σ, and $\forall t_1, t_2 \in T(\Sigma,X)$, if $t_1 \sim t_2$, then $\sigma(t_1) \sim \sigma(t_2)$.

Definition: A relation $\sim \subset T(\Sigma,X) \times T(\Sigma,X)$ is *Monotonic* if $\forall s, t_1, t_2 \in T(\Sigma,X)$ and $\forall$ occurrences u in s, if $t_1 \sim t_2$, then $s[u \leftarrow t_1] \sim s[u \leftarrow t_2]$.

Definition. The *Rewrite Relation* on a term algebra $T(\Sigma,X)$ for a term rewriting system R is the relation $\rightarrow_R \subset T(\Sigma,X) \times T(\Sigma,X)$ such that

(i) $l \rightarrow_R r$ for every $l \rightarrow r \in R$

(ii) $\rightarrow_R$ is closed under stability and monotonicity.

Notation. Given a relation $\rightarrow_R \subset T(\Sigma,X) \times T(\Sigma,X)$, we use the following notation.

$\rightarrow_R^*$ the transitive, reflexive closure of $\rightarrow_R$,
$\leftrightarrow_R$ the symmetric closure of $\rightarrow_R$,
$\leftrightarrow_R^*$ the symmetric, transitive, reflexive closure of $\rightarrow_R$.

Definition. A term t in $T(\Sigma,X)$ is *Reduced* by a rewrite rule $l \rightarrow r$ to a term t' if $\exists$ an occurrence u in t and a substitution σ such that $t/u = \sigma(l)$ and $t' = t[u \leftarrow \sigma(r)]$. A term t' is *Reachable* from t in R if $t \rightarrow_R^* t'$. A term t is in *Normal Form* with respect to R if it cannot be reduced by any rule in R.

We can use these definitions to say precisely what we mean when we say that a term rewriting system is equivalent to an equational theory.

Definition. A Term-Rewriting System, R for an equational theory, E is one which induces the same equivalence classes. That is $s \leftrightarrow_R^* t$ if and only if $s =_E t$.

When applying a term rewriting system we would like there to be a guarantee that a normal form can be found for all terms. We would like the rewriting to terminate, rather than to continue infinitely. This can be avoided if the rewrite relation has the following property.

Definition. A relation $\sim \subset T(\Sigma,X) \times T(\Sigma,X)$ is said to be *Noetherian* or *well-founded* if there are no infinite chains in the relation. ie no infinite sequences of terms of the form:

$$t_1 \sim t_2 \sim t_3 \sim \cdots$$

To ensure the termination of rewriting, we impose an ordering on the term algebra.

Definition. A *Termination Ordering* $>$ is a relation on $T(\Sigma,X)$ which is transitive, stable, monotonic, and Noetherian. Note that the Noetherian condition implies that $>$ cannot be reflexive.

If while generating rewrite rules, we take care to keep them embedded in the termination ordering, we can guarantee that the rewrite relation which results is also embedded in the termination ordering, and thus must be well-founded.

For the purposes of this paper, we will use a particular sub-class of termination orderings, outlined below.

Definition. A *Simplification Ordering* $>_t$ is a termination ordering on $T(\Sigma,X)$ such that $\forall t \in T(\Sigma,X)\ t >_t t/u$ for all non-root occurrences u (the *subterm property*).

Many termination orderings can be found in the literature. For examples and further discussion see [Ders87]. However, for the purposes of this paper we can ignore them further.

The Noetherian or well-foundedness condition is sufficient to ensure the termination of a sequence of rewrites of a term: by applying rewrites, a normal form of a term will always be found. However there is no guarantee that this normal form is unique. Another sequence of rewrites may produce a different normal form. To ensure the uniqueness of normal forms we need the rewrite relation to possess the following property.

Definition. A set of rewrite rules R (and the rewrite relation it defines) is said to be *(Globally) Confluent* if whenever a term s can be rewritten in two different ways, $t_1 \ {}_R^*\!\leftarrow s \rightarrow_R^* t_2$ then $\exists n$ *s.t.* $t_1 \rightarrow_R^* n \ {}_R^*\!\leftarrow t_2$. That is, both of these terms can be further rewritten to the same term. It is said to be *Locally Confluent* when $t_1 \ {}_R\!\leftarrow s \rightarrow_R t_2$, that is s is rewritten in two ways after a single step, then $\exists n$ *s.t.* $t_1 \rightarrow_R^* n \ {}_R^*\!\leftarrow t_2$.

1.3. ERIL

ERIL (Equational Reasoning - an Interactive Laboratory) is a configurable term-rewriting system designed by A.J.J.Dick at Imperial College, London and then subsequently at the S.E.R.C. Rutherford Appleton Laboratory for experimenting in the techniques of term-rewriting. The system provides facilities for the order-sorting of the terms used; a choice of the reduction ordering used upon the terms; a limited method for handling completion modulo equations; unfailing completion; and a range of facilities to

allow the interaction of various sets of axioms, rewrite rules and hypotheses. The use of a lattice of sorts gives additional reasoning power to the completion process. As such, the ERIL term-rewriting system is an ideal vehicle for experimenting with various strategies for using equational reasoning in general, and for theorem proving in particular. It allows the user to place rewrite rules, equalities and hypotheses in separate compartments, called equality sets, and then to choose how he or she wishes the various units to interact. For example, the user can choose to have a set of equations reduced by one set of rewrite rules, while not by another, by setting the "Reduced-by-set" attribute to the first set but not the other. Thus by placing the generated rules into differing sets (a process which the user will have to do himself; for details see the examples below) the user can control the extent by which the rules are rewritten and when. Further, the user can control which terms are superposed with which to allow the generation of new critical pairs.

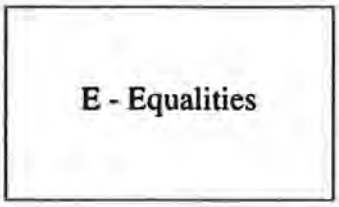
A variety of configurations for the ERIL system are described in the user manual [DicKal88], although not in terms of configuration diagrams (see below). This paper describes a series of configurations for Knuth-Bendix completion inspired by the work of Lescanne [Lesc89, Lesc90]. In addition, this paper contains some of the experiments carried out for the author's M.Sc. thesis [Matt88] where a number of configurations for theorem proving using ERIL are explored.

One point to note here is that the semantics of ERIL are based on a modification of the theory presented above as it based on Order-Sorted Algebra, whilst the simpler theory given is for the single-sorted case. However, in all the examples given the sort lattice is not exploited: all examples are single or at most multi-sorted, which is represented (by a simple extension) by the theory presented above. When restricted to single/non-intersecting sorts, the order-sorted theory collapses to the single/multi sorted case. This is not to say that the order-sorted method is not useful; indeed the sort structure can itself be exploited to control rewriting. However, that is beyond the scope of this paper.

2. Configuration Diagrams

Throughout this paper configurations will be described using a system of *Configuration Diagrams*, first introduced in [Matt88]. These diagrams describe the arrangement of equality sets and their interaction in a graphical fashion. These have then been used as the basis of the configurations of the ERIL system. The diagrams obey the following conventions.

A labelled box



represents a set of equalities, be they equations proper, rewrite rules or hypotheses. The boxes are connected by various arrows which represent the following.

Unbroken arrow (→)	Target set is reduced by source set
Dashed arrow (-->)	Result of superposing source set(s) is placed in target set
Dotted arrow (···>)	Elements of source set are moved into target set (possibly by hand)

Thus the configuration of the ERIL system is represented by an arrangement of boxes and lines giving the equality sets within the system and the interactions between them, which are entered into the ERIL system in terms of attributes, except for the move attribute which is given in commands. The ERIL system will then seek to maintain the attributes and will apply the attribute as soon as possible. Thus for example, if a set `A` is set to be rewritten by set `R`, then the rules in `R` will rewrite equalities as soon as they are entered into `A`.

The configuration is then driven by a command line. The syntax of that part of the ERIL command language used in this paper is given below.

Command	Action
m	Move equality from set given next, to a set given after choice of equality.
all	Pick every equation to act upon in turn from a selected equality set.
next	Pick the next equation in set in some prearranged order.

Thus for example given two sets `A` and `R`, the first equality in the set `A` is moved into set `R` by applying the command

```
m A next R .
```

The "next" equality is chosen according to some criterion set by the user such as the equality which has been in the set longest ("by-age" selection) or has the smallest number of operators in it ("by-size" selection).

3. Knuth-Bendix Completion.

A seminal paper in the development of the automation of equational reasoning was produced by Knuth and Bendix [KnuBen70] at the end of the 1960's. In this paper, they described a procedure for finitely generating a confluent set of rewrite rules for certain algebraic systems. The key idea of this paper was that to determine the confluence of the set of rewrite rules, it was sufficient to consider the "critical pairs" of the set alone.

> **Definition.** Given two rewrite rules $l_1 \rightarrow r_1, l_2 \rightarrow r_2$ such that for some substitution σ, $\sigma(l_1/u) = \sigma(l_2)$, that is the second rule superposes on the first at some occurrence u, then the *Critical Pair* is defined to be the pair of terms $(\sigma(l_1[u \leftarrow \sigma(r_2)]), \sigma(r_1))$. A critical pair is *trivial* if the normal forms of its two terms are identical and *divergent* if they are not. The set of critical pairs which can be generated by a set of rewrite rules R is denoted $CP(R)$.

Knuth and Bendix's observation was that if all possible critical pairs were trivial, then the rules set must be confluent, and further by the generation and testing of critical pairs confluent sets of rewrite rules could be generated of equivalent reasoning power to the

original axioms.

Knuth and Bendix present their algorithm in the form of a piece of English text. Others, for example [Forg84, HuOp80] give completion algorithms in terms of pseudocode. Both these are procedural descriptions of the algorithm, and prescribe one way of computing the completion. Bachmair [Bach87, Bach91], on the other hand, presents the algorithm as a set of inference rules that, together with a critical pair criterion, can be applied to a set of equations and a set of rewrite-rules. There is no control imposed on the order in which these rules are to be applied, apart from a general condition of "fairness" (see below). The inference rules are nondeterministic in their application.

There is a slight difference between the rules given below and those that Bachmair gives. He gives the side condition of KB2 as "*if* $s \,{}_R\!\leftarrow u \rightarrow_R t$". This is a very general condition which Bachmair restricts to a Critical Pair Criterion over and above the inference rules. Knuth and Bendix's criterion, of a critical superposition between left hand sides of existing rules, is embedded in the rules below rather than as an extra restriction on choice. This emphasises that you need only consider the critical pairs between existing rules to produce a canonical set of rewrite rules.

KB1 **Orienting an Equation.** An Equation can be oriented into a rewrite rule if its terms stand in the termination ordering.

$$\frac{(E \cup \{ s = t \}, R)}{(E, R \cup \{ s \rightarrow t \})} \quad \text{if } s >_t t$$

KB2 **Adding a new Equational Consequence.** New equations can be generated by finding critical pairs between the Rewrite-Rules.

$$\frac{(E\,, R)}{(E \cup \{ s = t \}, R)} \quad \text{if } (s,t) \in CP(R)$$

KB3 **Simplifying an Equation.** Equations can be simplified by applying the Rewrite Rules to their terms.

$$\frac{(E \cup \{ s = t \}, R)}{(E \cup \{ u = t \}, R)} \quad \text{if } s \rightarrow_R u$$

KB4 **Deleting a Trivial Equation.** Equations with identical terms are irrelevant and can be deleted.

$$\frac{(E \cup \{ s = s \}, R)}{(E\,, R)}$$

In addition to these four rules, which Bachmair calls Basic Completion, Bachmair also points out we can add two extra rules which can be applied to simplify rules by each other, which are essential if the procedure is to be practically implemented.

KB5 **Simplifying the left hand side of a rewrite rule.** If the left-hand side is reduced, it may not remain higher in the ordering than the right-hand side, and so the resulting equation is placed into E. Note that we cannot let a rule be rewritten by itself.

$$\frac{(E, R \cup \{ t_1 \rightarrow s \})}{(E \cup \{ t_2 = s \}, R)} \quad \text{if } t_1 \rightarrow_R t_2$$

KB6 **Simplifying the right-hand side of a rewrite rule.** If the right-hand side is reduced, it will remain lower in the ordering than the left-hand side, and so the resulting rule remains in R.

$$\frac{(E, R \cup \{ s \rightarrow t_1 \})}{(E, R \cup \{ s \rightarrow t_2 \})} \quad \text{if } t_1 \rightarrow_R t_2$$

Note that in the simplification rules, rewriting a rule by itself is excluded. On the right-hand side such a match is not possible in a termination ordering. On the left-hand side, the only match[2] can occur at the root of the term, in which case the rule would rewrite to the trivial equation.

In addition Dick [Dick87] includes another inference rule which deletes an equation which is completely subsumed by another, and is thus redundant.

KB7 **Deleting a subsumed equation.** If an equation e_1 is subsumed by another e_2, then when e_1 is oriented into an equation later in the process, it will be rewritten into an identity, and thus e_1 can be deleted at this stage.

$$\frac{(E \cup \{ s = t \} \cup \{ s' = t' \}, R)}{(E \cup \{ s = t \}, R)} \quad \text{if } \exists \sigma . \sigma(t) = t' \wedge \sigma(s) = s'$$

By giving the algorithm as a set of inference rules, we emphasise the freedom the user has in choosing an appropriate strategy for achieving completion. The inference rules can be applied in any order, or even in parallel. However, there is one important global constraint on the application of these inference rules: their application must be "fair".

Definition. A sequence of applications of inference rules is *fair* if and only if

(i) Every critical pair which can be generated, is generated.

(ii) Every critical pair generated is eventually either reduced to a trivial equation and deleted, or else converted into a rewrite rule.

---

[2] Whilst a left-hand side l cannot *match* with a proper subterm of itself l/u, that is $\nexists \sigma$ *s.t.* $\sigma(l) = l/u$, they can *unify* with one another, given a renaming of variables. That is $\exists \sigma$ *s.t.* $\sigma(l) = \sigma(l/u)$. In the later case, l can *superpose* with itself.

3.1. The ORME Approach to Strategies for Equational Completion.

Pierre Lescanne has exploited this freedom in the application of these inference rules. In [Lesc89, Lesc90] a series of different strategies are described for applying the inference rules for Knuth-Bendix completion. The differing strategies can then be compared for efficiency. These strategies are implemented as functions in the functional programming language CAML in his term-rewriting system ORME from a basic set of tools implementing the inference rules directly. Thus, for each strategy, a new piece CAML code must be written. The system is structured so that each of the basic operations of completion: rewriting; ordering; superposition etc, are available as "built-in" functions, which are then arranged together to form the strategy. This has the advantages of a large amount of reuse, and the strong formal basis which underlies functional programming languages, so proofs of the correctness of these new implementations can be given. However, knowledge of the CAML language is required to build new strategies, and proofs of the correctness must be built in the polymorphically typed λ-calculus which underlies CAML: a non-trivial process.

Strategies similar to those described by Lescanne, can be simply produced as configurations of the ERIL system. These configurations are now described and some results of their use given. It is claimed that they are complete strategies in that they implement at least the basic completion rules KB1-KB4 and use a fair strategy in the application of these rules.

Lescanne describes four general components of a completion procedure which he uses to construct strategies. These are a data-structure upon which the completion runs, a set of transition rules, a control of data-flow, and a toolkit of basic operations. These components have their analogues in ERIL. The data-structure in given by the separation of the domain of discourse into equality sets each of which has a particular function in the completion process. These have a set of attributes assigned to them which together with the built-in functions, such as identity checking and deletion, are parts of the toolkit. The transition rules, which describe the transference of equalities from one set to another, together with the control are given in the command language of ERIL.

3.2. A Naive Strategy

The most naive strategy is to have a strong separation in the use of the inference rules of the Knuth-Bendix system. So we take the current set of equations, orient them all, simplify (and delete as necessary) all rules and equations by these new rewrite rules, and then generate the critical pairs of these new rules with the existing rules of the system. This strategy is similar to Lescanne's N-strategy, and can reproduced within the ERIL system by using the configuration given in figure 1.

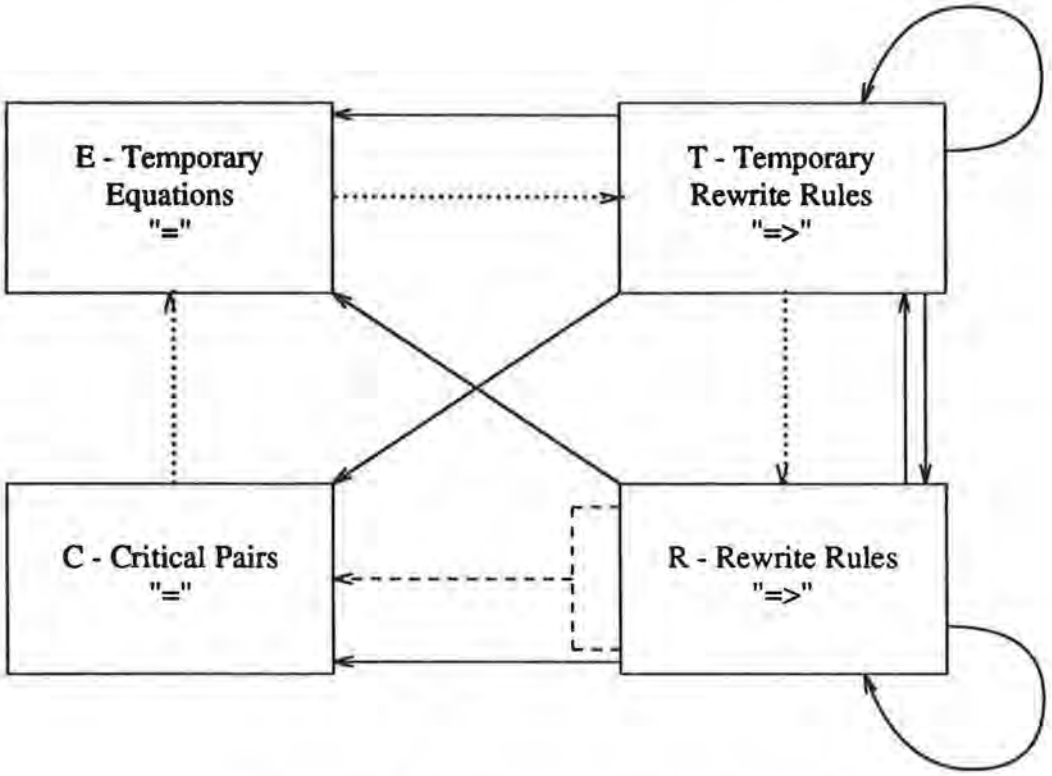


Figure 1: The N-Strategy - A Naive Approach.

This configuration is then used by placing the initial axioms into the set C and repeatedly using the command:[3]

```
m C all E m E all T m T all R
```

Thus all the current equations in C are first moved into E. This is to prevent the existing equations from being confused with any equations generated in this step of the completion process, particularly those resulting from reducing elements of set R, the current "considered" or "marked" set of rules, which have already had their critical pairs considered. The equations are then moved into T in order of size where they are oriented, and all other sets are reduced (on both sides) by them, with deletion as necessary. When all rules have been oriented (if possible), then they are moved into R where all critical pairs between the incoming rule and the existing rules in R are generated as the new rule is entered, and the new critical pairs are placed into C, to await consideration in the next cycle of the process.

[3] One of the disadvantages of the ERIL system is that multiple commands like this one cannot be repeated automatically. The user has to tediously repeatedly call the command. This may just be a matter of pressing a return key to activate the "repeat" mechanism in ERIL, but it is irritating to the user! However, this is nevertheless an automatic process as the user does not influence the running of the strategy once it has been set.

3.2.1. Correctness of the N-Strategy

It is claimed that the N-Strategy correctly and fairly implements at least the basic completion inference rules KB1-KB4 given above. An argument for correctness similar to those presented in [Dick87] is given. The various parts of the procedure implement the inference rules in a way corresponding to the description of the strategy below. Each section of the strategy is placed next to its corresponding inference rule. It must be observed that the data objects of the inference rules, **E** and **R**, are represented by the combined equality sets C ∪ E, and T ∪ R respectively.

Step	Action of Strategy	Associated Inference Rule
1	**All** equations moved from C to E.	No rule - an internal reorganisation of **E**.
2	Axioms selected from E and placed in T, and oriented into a rule.	KB1
3	New rule applied to existing rules in both R and T, resulting in reduced rules being placed in C.	KB5
4	New Rule applied to existing equations in C and E.	KB3
5	Repeat Steps 2 - 4 until E is empty, or unorientable equation generated.	–
6	Rule selected from T and placed into R.	No rule - an internal reorganisation of **R**.
7	New rule in R superposed on **all** other rules in R. Critical pairs placed in C.	KB2
8	Current rules applied to new critical pairs in C.	KB3
9	Trivial equations deleted from C	KB4
10	Repeat Steps 1 - 9 until C, E and T are empty, or unorientable equation generated.	–

This is a fair strategy as:

i). The selection of equations to be converted into rewrite rules is by the criterion of *distance* as defined in [Kuch83] (see L-Strategy below).

ii). All possible superpositions between a rule and all existing rules are generated on entering R.

3.3. A Simplifying Strategy

The naive approach above can be improved by observing that the critical pairs can be oriented as soon as they are generated and thus used for simplifying (and possibly deleting) the current set of rules before they are used for generating new critical pairs. Thus

we are interleaving the application of inference rules KB2 and KB3. This strategy can be reproduced in ERIL by the configuration given in figure 2.

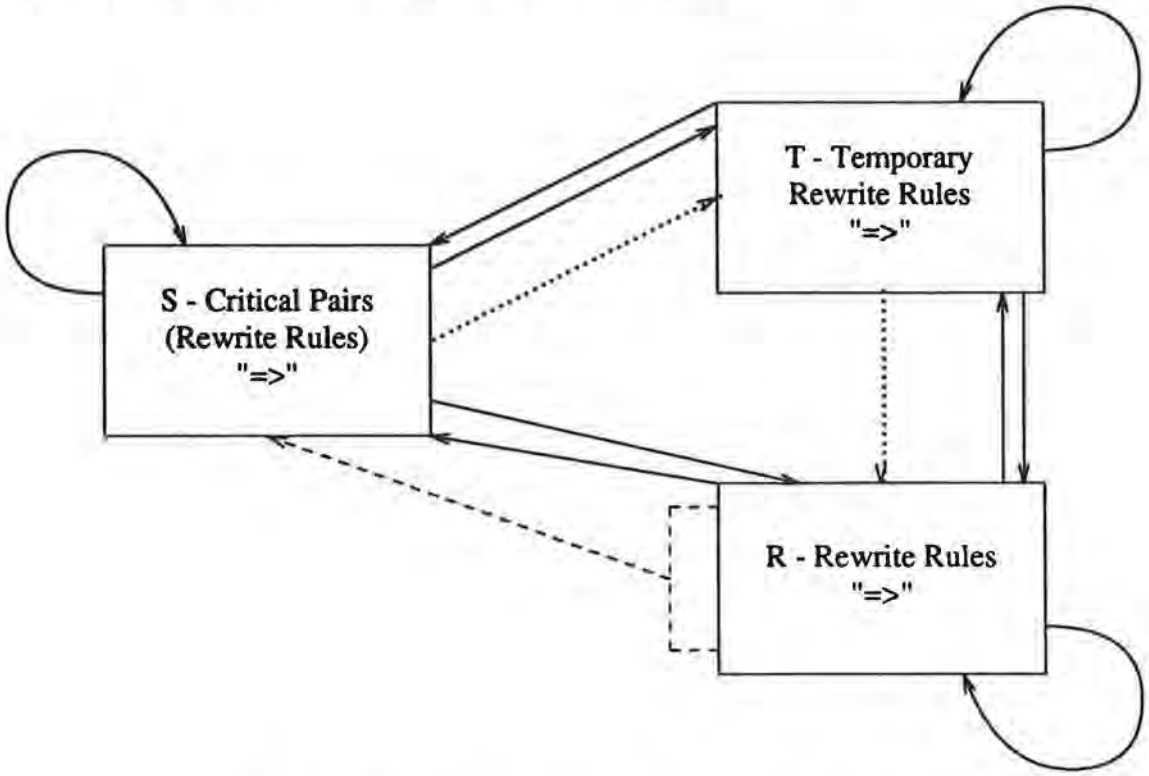


Figure 2: The S-Strategy - A Simplifying Approach.

This configuration is then driven with the repeated application of the command sequence

```
m S all T m T all R .
```

This strategy is similar to the N-strategy except that the critical pair set `S` has the equality type attribute set to rewrite rule, so as soon as critical pairs are created, they are oriented into rewrite rules. Then all the other sets of rules are simplified straightaway. If a rule in the set `T` is deleted, then no critical pairs are generated from it at all, whereas in the N-strategy above they would have been. There is no analogue of the set `E` in this strategy. Any rule in R would have already been reduced and placed in `S` by the time it is moved into `T`.

Again we give an argument for the correctness of this strategy.

Step	Action of Strategy	Associated Inference Rule
1	**All** equations in `S` oriented.	KB1
2	All rules in `S` reduced by `S+T+R`, and reoriented.	KB3, KB5, KB6, KB1
3	Trivial Equations removed from `S`.	KB4
4	All rules in `S` moved to `T`.	An internal reorganisation of R.
5	All rules in `T` moved to `R`.	An internal reorganisation of R.
6	New rules in `R` superposed on `R` and critical pairs placed in `S`.	KB2
7	Repeat Steps 1 - 6 until `S`, and `T` are empty, or unorientable equation generated.	–

This is a fair strategy for the same reasons as the N strategy.

3.4. A Least Critical Pair Strategy

Küchlin [Kuch83] considers a series of strategies for selecting the next equation for the generation of critical pairs. Both of the previously considered strategies have been variants of his *Breadth-first strategy*. Equations have been considered in cycles; that is, if we define a notion of *distance* for an equality as being 0 for a basic axiom and, for a critical pair, 1 + the maximum distance of its parent rules, then all rules of distance $n-1$ are selected for critical pair generation before those of distance n. We can vary this to instead select the next equation considered to be that of "least size". That is, select the rule which has the *least number of nodes on both sides* as the next rule to be considered for use as a reducing rule, and also for critical pair generation. The local selection strategy `by-size` within ERIL will pick equations using this criterion. We use this selection in conjunction with the configuration in figure 4.

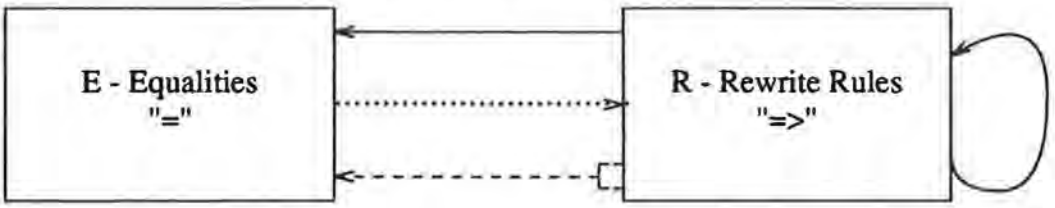


Figure 4: The L-Strategy - A Least Critical Pair Strategy.

This configuration is then driven with the application of the command sequence

```
m E all R .
```

Unlike the above strategies, this command sequence runs automatically without any

intervention from the user as when equations from `E` are entered into `R`, any critical pairs are placed back into `E` where they are selected by the still running "`all`" command above. As the current set of equations are not separated from the critical pairs as they are generated, when the system comes to choose the next equation to consider for reduction and generation of critical pairs, it chooses the "smallest" of them. Consequently a critical pair can be immediately selected and considered for critical pair generation itself.

We repeat the argument for correctness as in [DicKal88].

Step	Action of Strategy	Associated Inference Rule
1	Axioms moved from `E`, oriented and entered into `T`.	KB1
2	New Rule applied to left and right sides of all equalities. If reduced placed in `E`	KB3, KB5, KB6.
3	Trivial Equations removed from `E`.	KB4
4	New rule in `R` superposed on `R` and critical pairs placed in `E`.	KB2
5	Critical pairs reduced by `R`.	KB3
6	Trivial Equations removed from `E`.	KB4

3.5. Huet's Strategy

The last simplifying strategy is similar to that given by Huet in [Huet81]. The configuration of ERIL for this strategy is taken from [Dick87] where a demonstration of the correctness of this strategy is given.

The configuration (figure 3) is similar to that of the S-strategy (figure 2). The difference in this strategy is that the critical pairs are not immediately oriented into rules for reduction, so some redundant critical pairs are likely to be created.

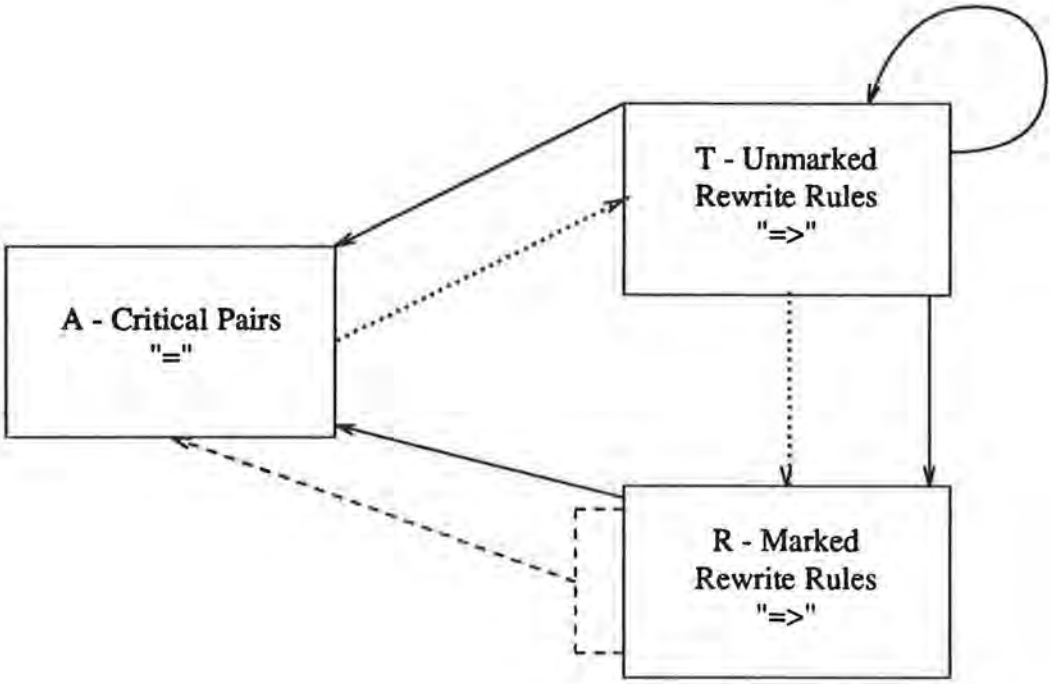


Figure 3: The H-Strategy - Huet's Strategy

This configuration is then driven with the repeated application of the command sequence

```
m A all T m T next R .
```

Unlike the S-strategy, only the first (under some fair selection strategy, such as by-age or by number of operators) rule in T is chosen. This would mean that, under a by-size selection strategy, the critical pairs of the smallest rules are always considered first, and as "small" rules are more likely to match in more equations, they are likely to reduce more strongly. However, more work is carried out attempting to match equalities by rules.

For completeness, we repeat the correctness argument given in [Dick87].

Step	Action of Strategy	Associated Inference Rule
1	Axioms moved from A, oriented and entered into T.	KB1
2	New Rule applied to left and right sides of all equalities. If reduced placed in A	KB3, KB5, KB6.
3	Trivial Equations removed from A.	KB4
4	Rule moved from T to R	An internal reorganisation of R.
5	New rule in R superposed on R and critical pairs placed in A.	KB2
6	Critical pairs reduced by R+T.	KB3
7	Trivial Equations removed from A.	KB4

3.6. A Reducing, Least Critical Pair Strategy

The L and S-strategies can be combined into one. By changing the attribute of the E set from equality to rewrite rule, the critical pairs are oriented as soon as they are generated and used to reduce the other rules in the system. This strategy is implemented by the configuration in figure 5.

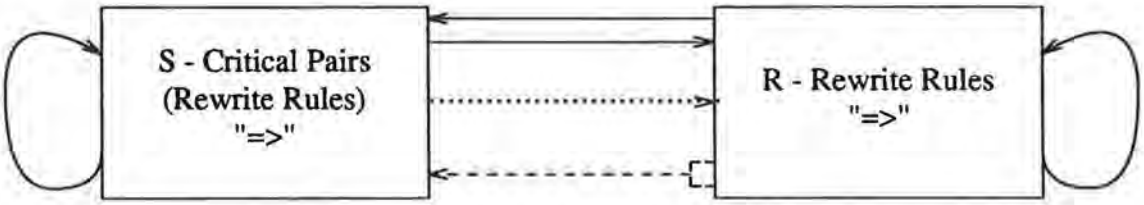


Figure 5: The SL-Strategy - A Reducing, Least Critical Pair Strategy.

This strategy is driven by the automatic command sequence similar to the L-strategy above.

```
m S all R .
```

This strategy combines the tactics employed in the S-Strategy and the L-Strategy, in that all critical pairs are immediately used as simplifying rules, and that the smallest of the current unconsidered rule set is the next for selection, regardless of when that rule was generated.

Again we give an argument for the correctness of this strategy.

Step	Action of Strategy	Associated Inference Rule
1	All equations in S oriented.	KB1
2	All rules in S reduced by S+R, and reoriented.	KB3, KB5, KB6, KB1
3	Trivial Equations removed from S, and R.	KB4
4	Next rule in S moved to R.	An internal reorganisation of R.
6	New rules in R superposed on R and critical pairs placed in S.	KB2
7	Repeat Steps 1 - 6 until S, and T are empty, or unorientable equation generated.	–

3.7. An Incomplete Strategy

An attempt to further improve the performance of the completion algorithm in the ERIL system is driven by the configuration shown in figure 6.

Figure 6: An Incomplete Strategy.

This configuration is driven by just entering the initial equations into the set R. As they enter, equations are oriented, and reduction take place. Then the next critical pair is generated between this and the existing rules. As critical pairs are generated, they are placed into R, and the previous critical pair generation is suspended while the critical pairs of the new rule are considered. This of course loops; we have a stack of suspended processes. This is a *depth-first* selection strategy in Küchlin's terms [Kuch83] and is incomplete.

3.8. Results of Experiments with the Configurations

The various configurations for Knuth-Bendix completion outlined above were tested on examples 1 (Group Theory I), 3 (Group Theory III), 6 (Group Theory IV), 10 (Loops), 12 ((l,r) Systems I), 13 ((r,l) Systems), 14 ((l,r) Systems II), 16 (Central Groupoid II) and 17 (Central Groupoid III) from Knuth and Bendix's original paper. These were chosen as non-trivial examples of completion of algebras, which do not involve the addition of extra operators at a later stage in the completion procedure when unorientable critical pairs are generated. The results can be compared in terms of the number of critical pairs

generated in the completion process. These results were obtained by running ERIL with an extended heap size (up to 8Mb), using a Knuth-Bendix ordering, either by the user setting the weights, or the incremental Knuth-Bendix ordering based upon [DicKalMar88]. The incremental ordering is only practical on small examples; on larger ones it overflows the heap.

Example	N-Strategy	S-Strategy	L-Strategy	H-Strategy	SL-Strategy
Group Theory I	338	85	100	80	80
Group Theory III	1501	212	112	96	95
Group Theory IV	617	112	117	92	91
Loops	30	30	32	30	30
(l,r) Systems I	820	192	336	171	169
(r,l) Systems	* (a)	280	171	300	290
(l,r) Systems II	– (b)	520	631	606	591
Central Groupoid II	870	521	469	425	400
Central Groupoid III	292	292	292	292	292

(a) In the (r,l) Systems problem, the N-strategy generated the equation

$$-(-x_3 + x_1) + (x_3 + x_2) = -(x + x_1) + (x + x_2)$$

This equation has variables on each side which do not occur in the other and consequently is unorientable. This equation is not generated in the other strategies.

(b) The N-strategy filled the available heap in the case of the (l,r) Systems II problem (after some 2289 critical pairs). This strategy suffers from generating large numbers of critical pairs, which are subsequently deleted, before being oriented, by other equations being moved into the T set. They have to be stored in the mean

time and, as a consequence, the heap space rapidly fills in a large problem.

It can be seen from these tests that the N-strategy is the least efficient of these strategies and the SL-strategy the most successful (but not universally - see (r,l) Systems for instance). Huet's strategy is also a fairly efficient.

4. Equational Reasoning in Horn Clauses

We now turn our attention to the methods employed in using equational reasoning for automated theorem proving. Several methods have been developed to apply equational reasoning to theorem proving, either of hypotheses of equational theories, or to apply completion to First Order Predicate Calculus. These methods usually make modifications to the basic Knuth-Bendix completion algorithm appropriate to the specific problem domain. We first direct our attention to Paul's method for reasoning with equality in Horn Clauses.

Paul [Paul85b] presents a method for carrying out equational reasoning within theories which are defined by Horn clauses with equality rather than by equational theories. Thus he is attempting to produce an alternative to the paramodulation methods explored by the resolution theorem proving community (for paramodulation see [Love78]). He does this with a modification of the Knuth-Bendix algorithm. He presents two versions of his method, one of which is more general but unfortunately fails to be Noetherian, the other being restricted to Horn clauses but is canonical.

4.1. The Method

Both of Paul's methods depend on the following observation proved in [Paul84].

> **Theorem:** If S is a theory defined by a set of (implicitly universally quantified) clauses $\{C_1,...,C_n\}$ and F is a universally quantified formula then F is a valid consequence of S in first order logic if and only if the equation $F = TRUE$ can be deduced by equational reasoning from $S' = \{C_1 = TRUE,...,C_n = TRUE\}$ together with BOOL, where BOOL is a complete equational axiomatisation of the Boolean calculus.

Paul notes that such an axiomatisation does indeed exist and quotes the work of Hsiang [HsiDer83] which presents an axiomatisation confluent under Associative/Commutative unification. We can also use this technique as a refutational proof technique. If F is not satisfiable by S then we can use the above technique to reduce F to $FALSE$, resulting in $TRUE = FALSE$, a contradiction.

If, further, we introduce an explicit equality predicate EQ in the object language and the axioms of equality $AX(EQ)$ defined in equational Horn clauses:

Reflexivity	$EQ(x,x) = TRUE$
Symmetry	$\neg EQ(x,y) \vee EQ(y,x) = TRUE$
Transitivity	$\neg EQ(x,y) \vee \neg EQ(y,z) \vee EQ(x,z) = TRUE$
Congruence:	
in functions	$\neg EQ(x,y) \vee EQ(f(x_1,...,x,...,x_n), f(x_1,...,y,...,x_n)) = TRUE$ for any function f of arity n in any argument
in predicates	$\neg EQ(x,y) \vee \neg P(x_1,...,x,...,x_n) \vee P(x_1,...,y,...,x_n) = TRUE$

for any predicate P of arity n in any argument

then we can say that

> **Theorem:** F is a E-valid formula of S that is valid in any model where EQ is interpreted as identity, if and only if $F=TRUE$ can be deduced from $BOOL \cup AX(EQ) \cup S$.

Thus to prove $EQ(t,s)$ is a valid equality under S, we run completion on $BOOL \cup AX(EQ) \cup S$ and attempt to reduce $EQ(t,s)$ to $TRUE$. Unfortunately this procedure proves to be of limited use, as $BOOL \cup AX(EQ)$ does not transform into a canonical rewrite system. Completion generates non-orientable permutative critical pairs such as

$$EQ(x,y) \wedge EQ(x,z) = EQ(x,y) \wedge EQ(y,z).$$

Paul is only interested in problems involving the equality predicate alone, while the above method gives a general method for solving any universally quantified formula.

4.2. The Second Method

Paul subsequently modifies this method turning his attention to theories defined by Horn clauses alone, and queries which are just a single equality predicate. He presents a confluent rewrite system R_0 which he claims is all we need to know about the boolean calculus and the axiomatisation of equality.

$$\begin{array}{rcl} \neg TRUE & \rightarrow & FALSE \\ \neg FALSE & \rightarrow & TRUE \\ X \vee TRUE & \rightarrow & TRUE \\ TRUE \vee X & \rightarrow & TRUE \\ X \vee FALSE & \rightarrow & X \\ FALSE \vee X & \rightarrow & X \\ EQ(X,X) & \rightarrow & TRUE \end{array}$$

To solve the equality problem for a theory S defined by a set of Horn clauses $\{C_1,...,C_n\}$, we run the Knuth-Bendix completion algorithm over R_0 and the set of equations defined as $\{C_1 = TRUE,...,C_n = TRUE\}$ together with the extra inference rule:

> **Collapsing the Equality Predicate.** If an equality is equal to $TRUE$ in the object language, assert it as a new axiom.

$$\frac{EQ(t,s)=TRUE}{t=s}$$

Thus Paul avoids the need to have the axioms of equality explicitly in the equation set: they are dealt with by the built-in underlying theory governing the use of the meta-level operation "=". If the equation TRUE = FALSE is generated, then S is unsatisfiable.

4.3. Configuring ERIL for Horn-Clause Theorem Proving.

We can use Paul's method for Horn-clause theorem proving within ERIL. This is illustrated in [DicKal88]. However, as the Knuth-Bendix Completion procedure is run naively to prove theorems in either the straightforward constructive manner for equational theories or in any of the refutational variations discussed above, the proof so

generated may not be the shortest proof possible for the problem in hand. Thus there is potential for improving the performance of the Knuth-Bendix completion procedure by choosing carefully which classes of equalities to convert into rewrite rules, and which rewrite rules to generate critical pairs from in various circumstances. The strategy chosen in solving a particular problem can affect the efficiency of the proof to a great extent. Therefore, we explore various configurations of the ERIL system for the two theorem proving methods introduced earlier.

4.4. A Basic Configuration for Paul's Method

The use of ERIL in a non-terminating context can best be illustrated by considering the configuration of the ERIL system required for Paul's second, more restricted method for investigating the equality problem in a Horn-clause logic context.

In Paul's [Paul85b] paper he presents a run through of his theorem-proving method. To carry out this method, we configure the ERIL system as in figure 7.

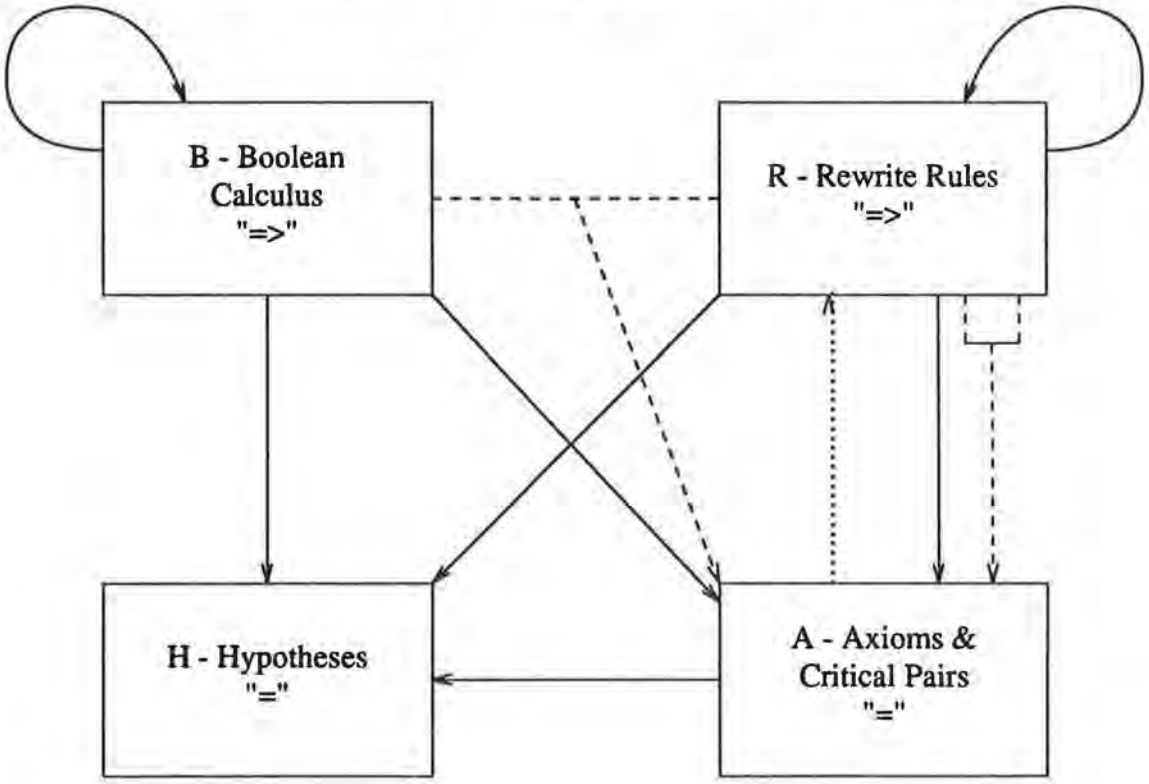


Figure 7: A Basic Strategy for Paul's Method.

This configuration is explained as follows. There are four sets of equations.

B - Boolean Calculus	The set of rewrite rules given earlier which describe the subset of the Boolean calculus needed for Horn-Clauses.
R - Rewrite Rules	The Rewrite rules which are derived from the equational theory under consideration for the problem

domain. Note that this set is not necessarily canonical.

A - Axioms & Critical Pairs — Axioms of the underlying equational theory. All critical pairs are also entered into this set.

H - Hypotheses — Queried equalities are placed inside this set. Note that they may or may not be in a negated and Skolemised form depending on whether constructive or refutational theorem proving is being performed.

These four sets are linked together in the following fashion.

Critical pairs can be generated in two ways. Either between members of the current rewrite rules set R, or between rewrite rules of the Boolean Calculus B and the current rewrite rule set R. Note that since B already is a canonical rule set we do not need to consider superpositions between members of this set since there will only be trivial ones (ones which reduce to x = x). All critical pairs generated by either means are placed in the axiom set, A. The rewrite rules, R are interreduced, and the Axioms are reduced to their current normal form by using both the Rewrite rules R and also the Boolean Calculus, B. The three sets B, R, and A reduce the set of hypotheses H. Only two of these sets are rewrite rules; ERIL allows the user to reduce by a set of equations. In this case ERIL checks whether any equation in H is subsumed by, ie is an instance of, an equation in A.

Note that in the above diagram a reduction link (a solid arrow) from B to itself is included. This is strictly superfluous as a successful reduction will never take place. This set is left entirely unchanged and only acts on other sets. This is included here as it is in the configuration described in the user manual.

All operator symbols in ERIL have to have an associated sort. The atomic propositions and boolean connectives used in the examples are declared to be of sort "`bool`". A feature of the ERIL system is that when an equality of the form

$$(t_1 = t_2) = T$$

is generated, where T is the boolean constant "true" and the first "=" is being used as the equality predicate, then ERIL will automatically collapse it to the equation

$$t_1 = t_2$$

Thus we can satisfy Paul's modification to the Knuth-Bendix algorithm.

To illustrate the power of this configuration, and to provide a benchmark for later configurations to be considered, we give the example of Henkin models ran through by both [DicKal88] and [Paul85b].

A Henkin model is a set S with constants "0" and "1", operator "_/_", and predicate LE(_,_) which obeys the following axioms.

$$LE(x, 1)$$
$$LE(0, x)$$

$$LE(x/y,x)$$
$$EQ(x/y,0) \supset LE(x,y)$$
$$LE(x,y) \supset EQ(x/y,0)$$
$$LE(x,y) \wedge LE(y,x) \supset EQ(x,y)$$
$$LE((x/y)/(z/y),(x/z)/y)$$

These axioms are converted to Horn-clause form (that it as a set of disjunctions of positive and negative literals), then converted into equations by setting each axiom equal to T (true), and then entered into the axiom set A above. The boolean calculus for Horn-clauses given above is entered into set B. Paul gives seven hypotheses he wants to show, and Dick and Kalmus give both constructive and refutational proofs methods for solving them. The first four hypotheses are proven constructively. They are placed together in set H, and axioms and subsequently generated critical pairs are moved from set A to set R, with the critical pair generation process automatically producing new equations, and the reduction mechanism keeping the rewrite rules, axioms and the hypotheses in normal form. The `by-size` selection strategy is used to pick the next rule for consideration. No canonical set of rewrites is created, but the system signals when the hypotheses have been reduced to identical terms on both sides, and are thus proven.

The results of running this configuration on these first four hypotheses are as follows. We give the number of critical pairs that ERIL generates, and also the number of rewrite rules that are created and kept in the course of the proof. The results are cumulative; they were proved one after the other and each used as lemmas for the next, and the critical pair counts were added together. For comparison we give the number of rules which Paul claims he used to prove each of these propositions. He in turn compares his results with the same problems proved on a Resolution/Paramodulation theorem-prover and shows that the equational method is more successful in terms of the number of clauses generated and kept (the equivalent to critical-pair generation in the equational-reasoning context). ERIL seems to be more successful than Paul's strategy. It must be noted that Paul's figures do not include the rewrite rules which are derived directly from the axioms; they are included as the initial members of the set of rewrite rules. In the ERIL strategy only the Boolean calculus is initially set as rewrite rules; the Henkin axioms are entered one by one into *R* using the preset "smallest equation" strategy. Thus the figures below for ERIL include the directly derived rules.

Hypotheses	**ERIL**		**Paul's** (rules)
	Critical Pairs	**Rules**	
$x/1 = 0$	6	5	1
$0/x = 0$	6	5	2
$x/x = 0$	121	17	30
$x/0 = x$	359	25	66

The next theorem that Paul attempts to prove is the transitivity of the LE predicate. That is

$$(LE(x_1,x_2) \wedge LE(x_2,x_3)) \Rightarrow LE(x_1,x_3)$$

This both Paul and Dick and Kalmus solve by a refutational approach. This first requires

the implicitly universally quantified formula to be negated and skolemised. This results in the three conjuncts

LE(A,B)
LE(B,C)
¬LE(A,C)

where A,B, and C are Skolem constants. We put these three formulae into the set of axioms (set equal to T of course) with the other axioms, and the four theorems we have already proved above, which we can use as valid lemmas. Other than these lemmas we start the completion process from scratch.

For this refutational method of theorem proving we do not use the Hypotheses set. The goal clauses which we wish to refute are included in the set of axioms. The derivation of a refutation is detected in ERIL as a by-product of the order-sorted mechanism in the heart of the ERIL system. The sort **bool** of truth values has two disjoint subsorts **true** and **false** each with only one constant **T** and **F** respectively. If the completion process derives the formula **T** = **F**, then the sorting mechanism detects that the equality is trying to equate two members of incomparable sorts, and thus informs the user of this and displays how the contradiction came about, that is displays the proof. Thus the refuted clauses have thrown up a contradiction and thus are unsatisfiable, so the original proposition holds.

Paul gives two other problems to solve.

$$LE(x/y,z) \Rightarrow LE(x/z,y)$$
$$LE(x,y) \Rightarrow LE(z/y,z/x)$$

which he solves by the refutational method. Dick and Kalmus do not give the proofs of these in their manual. We cannot use the rewrite rules generated in the proof of the previous, transitivity result, as the axiom set contains the contradiction $T = F$. So we have to start again from scratch. However in each proof, we can assume that the previous result is true, and include it in the axiom set. The formulae Skolemise into the pair of conjuncts

$$LE(A/B,C), \neg LE(A/C,B)$$
and
$$LE(A,B), \neg LE(C/B,C/A)$$

Again we compare the results below with those of Paul. At first sight the ERIL strategy seems to be significantly worse than Paul's, but it must be remembered that Paul's figures do not include the directly derived rules, while the ERIL figures do. As there are some fourteen initial axioms at this stage, (we include the previously derived lemmas and the hypothesis) the results for ERIL and for Paul seem roughly comparable.

Hypotheses	ERIL		Paul's (rules)
	Critical Pairs	Rules	
Transitivity of LE	273	35	20
$LE(x/y,z) \Rightarrow LE(x/z,y)$	306	32	25
$LE(x,y) \Rightarrow LE(z/y,z/x)$	Filled Heap Space		147

4.4.1. Observations

In this refutational strategy, the goal is involved in the theorem proving process. However, it cannot be really said to be goal-directed. The goal interferes with the completion process to form a contradictory equational theory. But the way the completion process proceeds is not directly influenced by the goal and still potentially may go down blind alleys. So it may be called more properly a *goal-influenced* strategy. We would like to be able to allow the goal to influence the choice of the next move more directly.

Another criticism of this refutational strategy is that when a query has been proven, all the current rewrite rules and equations have to be thrown away as they form an inconsistent set. So to prove another query from the same problem domain we have to start all over again, despite the fact that many rewrite rules derived from the Boolean calculus and the Henkin axioms and proven lemmas alone are valid rewrite rules for this theory. It would save much recalculation if we could save the valid rewrite rules and just throw away the inconsistent ones resulting from the negated query.

4.5. A New Theoretical Strategy

To overcome the difficulties encountered in the standard ERIL strategy outlined above a new strategy proposed. This strategy comprises of two parts. One is a new configuration of the ERIL system to accommodate a more flexible approach to theorem proving process. The second part is the selection strategy superimposed upon this configuration to exploit its flexibility to the full.

4.5.1. The Configuration

The new configuration is best expressed in the diagram below.

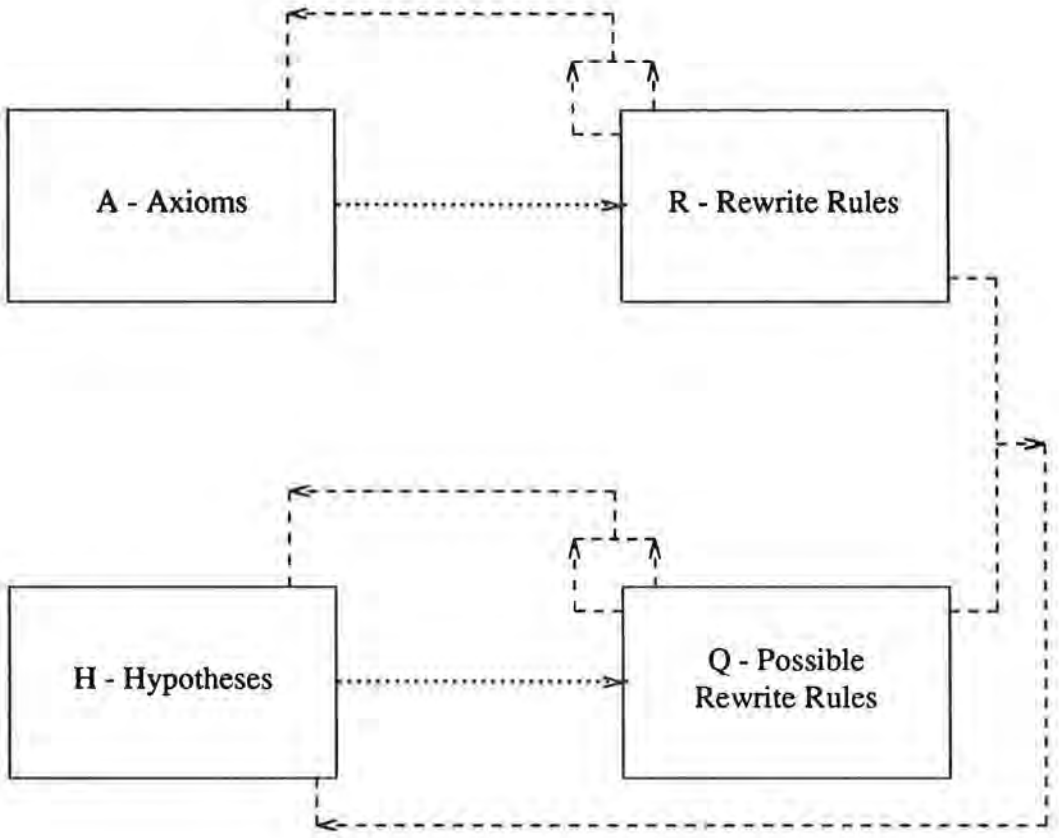


Figure 8: A Configuration for Paul's Method.

In this strategy we have four sets as follows

A - Axioms	The set of axioms of the theory in equational form. The initial set of domain axioms is placed in this set. Critical Pairs which are known for certain to be valid consequences of this theory are also placed in this set.
R - Rewrite Rules	Rewrite Rules which are known to be definite valid rewrite rules of the equational theory are placed in this set.
H - Hypotheses	Queried equalities are placed inside this set. These are placed in this set in a negated and Skolemised form as refutational theorem proving is being performed. Also placed in this set are those critical pairs generated as a result of the hypotheses being posed.
Q - Possible Rewrite Rules	Place rewrite rules which are of limited use. If the hypothesis in set H is proven valid, then the use of this set of rewrite rules will lead to a contradiction.

Axioms of the equational theory are moved into R and oriented into rewrite rules. Any critical pairs formed by superposing members of R together during the process are placed back into A. These are universally valid rules and equational consequences of the theory. This part of the process is just like ordinary Knuth-Bendix Completion.

The negated and Skolemised hypotheses placed in H can be moved into Q and oriented into rewrite rules. Remember that one query can result in a set of equations after negation and Skolemisation. Elements of Q can superpose with one another, and the resulting critical pairs, which are consequences of the hypotheses are placed back into H. In addition we want the equational theory to be applied to the hypotheses. This achieved by allowing superpositions to take place between elements of R and Q, the resulting Critical pairs being placed back into H. This process is like the generation of subgoals.

In the above diagram the reduction arrows are omitted for the sake of clarity. However for the process to work efficiently sets A and H should be reduced by the valid rewrite rules, R, and in addition set H should also be reduced by the set of hypothetical rewrite rules, Q.

4.5.2. The Selection Strategy

In order to maximise the goal direction possible in this configuration, we impose the following actions upon it, in order of priority.

(1) Move $h \in H$ into Q and orient into a rewrite rule.

(2) Superimpose elements of Q and place the resulting Critical pairs into H.

(3) Carry out superpositions between elements of R and Q and place the resulting critical pairs in H.

(4) Carry out superpositions between elements of R and place the resulting critical pairs in A.

(5) Move $a \in A$ into R and orient into a rewrite rule.

Each step is carried out until there is nothing more it can do unless it is possible to do something with a higher priority, which takes precedence and thus interrupts. Thus, for example, if in the process of generating critical pairs between R and Q we generate a non-trivial critical pair in H, then we should straight away find the consequences of this sub-goal and move it into Q, generate critical pairs in Q and repeat until no more critical pairs are generated in Q. Then we can return to the generation of critical pairs between R and Q. When no more critical pairs are generable between R and Q, then we return to generating critical pairs in R, and then to moving from A into R to apply more and more of the theory to the theorem proving process. The hypothesis is shown to hold when $true = false$ is generated in the set H. Thus the negated, Skolemised version is shown to lead to an inconsistency, and thus the hypothesis must be correct.

Thus by applying this theorem proving strategy, it is hoped that the goal and the consequences or subgoals derived from it play a major role in the theorem proving process. At all times the adding of consequences which derive directly from the goals take precedence over other consequence of the theory. And also as soon as some consequence of the theory is introduced, its implications to the goal are explored as fully as possible at the current stage, before more consequences are derived. Thus the strategy is more

goal-directed than the previous method, and goals should be proven more efficiently.

Another advantage of this new strategy, is that the valid rewrite rules and critical pairs are kept separate from the rules and critical pairs resulting from the inconsistency. Thus these equalities are of universal validity, and so when we wish to solve another hypothesis, we do not need to throw all our previous work away. We need only delete the elements in sets H and Q. We can keep the equalities in sets A and R and use them in the proof of the next theorem, thus saving much work on partial results.

4.6. The ERIL Implementation

The above theoretical strategy has to be modified if the ERIL system is to perform the moves automatically. The problem lies in ERIL's `link` attribute. This attribute is only set globally for each set. This means that whichever set the elements of the current set are superposed with, the resulting critical pairs are always placed in the same set. Thus in the new configuration proposed above, a `link` attribute of *R* set to *A* would result in both the critical pairs generated by superposing *R* on *R* being placed in A, and also those generated between *Q* and *R*. However, we know that the rules in *Q* are of dubious validity, while the critical pairs generated between members of *R* are valid consequences of the theory, so we wish to keep the two classes of critical pairs separate. The critical pairs generated between *R* and *Q* should be placed back into *H*. However, the only method in the current ERIL architecture to allow this distinction is a tedious set of commands where we specify which rule is superposed with which on an individual basis and where the resulting critical pair(s) are to be placed. This is using ERIL at its finest granularity of interaction and it turns out to be a very tedious process. We would like some degree of automation for the more routine and mechanical aspects of the completion process, while retaining some control over the system in the choice of configuration and strategy.

To overcome this difficulty, we adopt the architecture as shown in figure 9 below. We introduce a new set *T*, which contains a copy of the rewrite rules normally contained in *R*. The `suPerpose` attribute can then be set between this set and *Q* to generate critical pairs, which are then placed by the `link` attribute into *H*. No critical pairs are generated between elements of *T* alone. The set *R* now only generates critical pairs between its own members, which are placed in *A*. Thus we have separated the universally valid consequences of the theory, and the consequences of the (presumably false) goal.

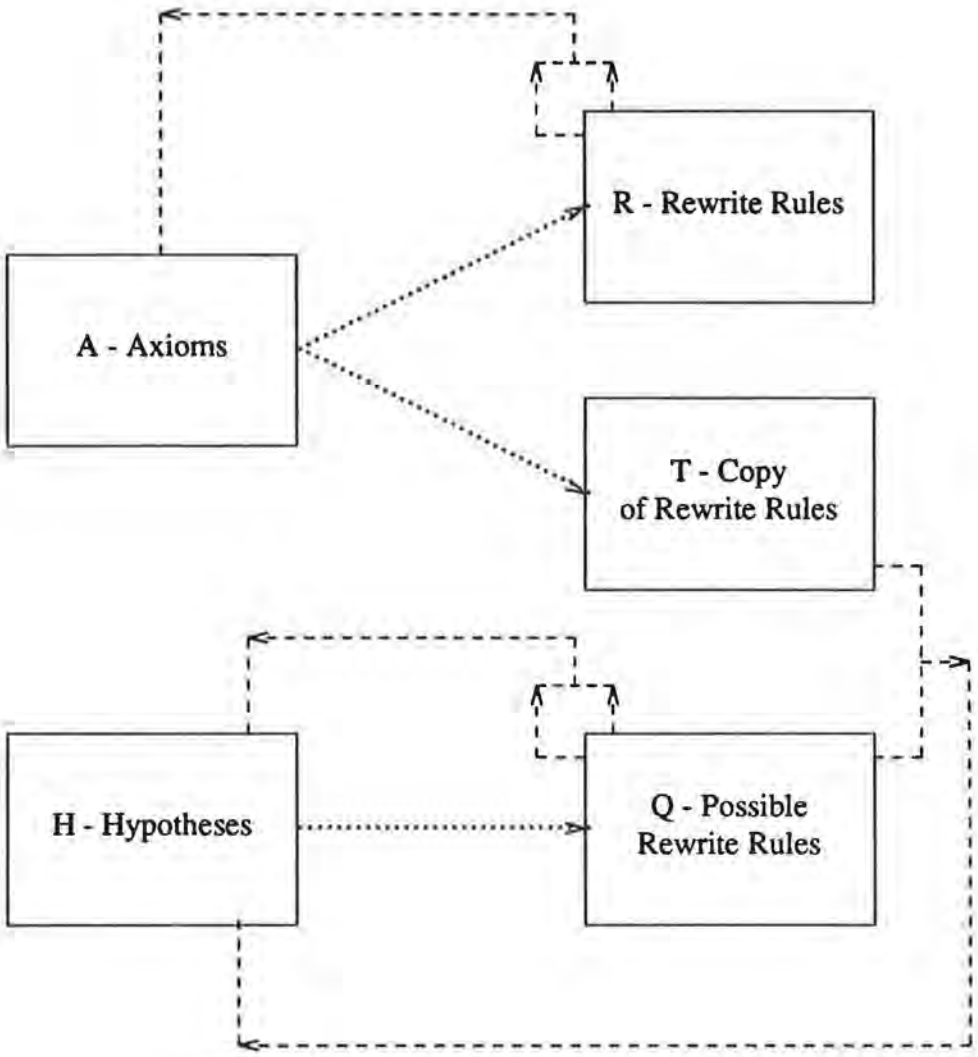


Figure 9: The ERIL Implementation of the Configuration.

However even this configuration presents a difficulty: there is no mechanism in the existing ERIL system for moving an equality simultaneously into two sets at once. So we cannot keep R and T identical at all times. This means we have to introduce the axioms into T and R at different times. This difficulty can be turned to our advantage as it gives us the ability to choose which order to introduce rules into R, and thus which critical pairs are generated. We have an extra dimension to our selection strategy.

4.7. An Example using the New Configuration

To illustrate how this new configuration works, we again use Paul's example of the Henkin model. The results should be compared with those Paul obtained and with the results using the standard ERIL configuration given earlier.

Within the actual configuration given above we do have different options in choosing a strategy for selecting equations to be converted into rewrite rules.

4.7.1. Strategy 1

This strategy proceeds in the following order.

1 Move Hypotheses from H to Q and orient

2 Form critical pairs between elements of Q, placing any new critical pairs back into H, then repeat steps 1 and 2 if applicable.

3 Move an axiom from A to T. Form critical pairs between T and Q, placing any new critical pairs back into H, then repeat steps 1 and 2 if applicable.

4 Move the same axiom into R and create critical pairs between elements in R. Critical pairs are then placed back into A.

In this strategy, we keep the sets T and R much the same. There can only be a maximum of one extra rule in T which is not in R.

The introduction of the rewrite rules into R had to be done by hand. The command `m A next T` deletes the axiom from A and orients it in T. A similar command `m T next R`, would delete the rule from T. There should be a facility in ERIL to `copy` equalities from one set to another. However in the current version this command does not work correctly. Consequently the more tedious procedure of using the `load` command had to be applied to enter the rule into the correct set.

The results of using this strategy on the Henkin Model example are as follows. Note that the Boolean calculus is in both the set *R* and the set *T*, placed there on initialisation. The linking system again prevents us from having the Boolean calculus in a separate set, as it cannot form critical pairs with both sets and put them in different places.

Hypotheses	ERIL			
	Critical Pairs	Rules		
		T	R	Q
$x/1=0$	5	14	13	1
$0/x=0$	7	14	13	1
$x/x=0$	117	29	27	3
$x/0=x$	372	47	33	2
Transitivity of LE	517	47	33	15
$LE(x/y,z) \Rightarrow LE(x/z,y)$	Filled Heap Space.			

There is a slight improvement here over the previous results. Unfortunately the large number of critical pairs created and the duplication of two sets of identical rules meant that the system ran out of heap space in the solution of the last two problems.

There are some comments to be made. Firstly the figures given above are cumulative. So for instance, there are 255 extra critical pairs generated between proving x/x = 0 and x/0 = x. Secondly that despite appearances to the contrary, the above figures do not result from running the above strategy, which should keep the *T* and the *R* sets the same. This is because the elements of R were subsumed and deleted, while those in T were not. This is a slight discrepancy, but it is not believed that it affects the overall result. Lastly, a large number of unnecessary critical pairs are created, particularly between elements of *R*. This means that extra elements are being added to the set A before all the effects of

the last set of critical pairs have been considered by adding them to T. This suggests we use the following strategy.

4.7.2. Strategy 2

This strategy proceeds in the following order.

1 Move Hypotheses from H to Q and orient

2 Form critical pairs between elements of Q, placing any new critical pairs back into H, then repeat steps 1 and 2 if applicable.

3 Move all axioms from A to T. Form critical pairs between T and Q, placing any new critical pairs back into H, then repeat steps 1 and 2 if applicable.

4 Move a rule in T, but not in R, into R, and create critical pairs between elements in R. If any non-trivial critical pairs are generated, they are placed back into A, and step 3 is repeated.

This strategy does not maintain the identity of T and R. Any critical pairs generated between members of R are applied immediately to the hypotheses in Q by moving them all into T. Only when A becomes empty do we consider adding more elements to the set R. Thus we delay generating critical pairs between elements of R as long as possible.

The results of using this strategy on the Henkin Model example are as follows. We use a `by-size` selection criterion. While again this strategy is fair (Küchlin [Kuch83] points out that there are only a finite number of equalities possible of a certain length in a finite language and so choosing by size will mean that every critical pair will eventually be considered), it proved extremely tedious in the current ERIL architecture. It required close inspection of the equation sets to determine which rules was the smallest rule in T which was not in R.

Hypotheses	ERIL			
	Critical Pairs	Rules		
		T	R	Q
$x/1 = 0$	2	14	9	1
$0/x = 0$	4	14	9	1
$x/x = 0$	117	35	28	2
$x/0 = x$	297	48	34	1
Transitivity of LE	618	48	34	10
$LE(x/y,z) \Rightarrow LE(x/z,y)$	Filled Heap Space.			

This does seem to be something of an improvement. The number of critical pairs needed to prove $x/0 = x$ has dramatically decreased for example. There seem to be more equations in R however. This is probably because some rules which are in T would actually reduce or subsume elements of R. but have yet to be entered.

Results for the last three problems are still disappointing. Large numbers of critical pairs are being created, but we do not seem to be getting any nearer to the solution. Also, frustratingly, there are elements in A which would help reduce the problem, but because of our strategy which tells us to keep on moving elements between H and Q until H is

exhausted, we are prevented from applying them as there is no apparent end to the critical pairs which are generated between Q and T, and Q and Q. We are running against the fairness condition here. The next problem is how to devise a fair strategy so we can consider the pairs generated between R for instance, while still retaining the high degree of goal influence in the new strategy.

4.7.3. Strategy 3

We apply the following criteria to the selection of the next equation to consider.

> Choose the smallest available element from either A or H to T or Q respectively. If A is empty, choose the smallest of T or H to move to R or Q respectively.

The results of using this strategy are as follows. The first four problems have exactly the same results as last time. The remaining problems give the following results. The number of critical pairs given is the total number of extra critical pairs for that particular goal.

Hypotheses	ERIL			
	Critical Pairs	Rules		
		T	R	Q
Transitivity of LE	233	43	32	8
$LE(x/y,z) \Rightarrow LE(x/z,y)$	200	44	33	8
$LE(x,y) \Rightarrow LE(z/y,z/x)$	Filled Heap Space			

There is one alteration to the strategy rules here; is was felt legitimate to add the transitivity rule straight into the set T.

This has proved a great improvement. The number of critical pairs needed to prove the transitivity law has decreased dramatically, and the sixth problem has proved soluble for the first time and using less critical pairs than the Dick and Kalmus configuration.

4.8. Further Strategies

Two more selection strategies also could be simply devised, one incomplete, and the other which is believed to be fair.

The incomplete strategy is the arbitrary size limit which Paul uses for some of his results. He rejects any equality which is longer than 14 in length where each function or variable symbol counts one. He admits that this is incomplete, but it does indeed work for this example, and also reduces the work required. It would be interesting to try to integrate this method into the new configuration. It would require a condition on membership on a equation set. The oversized equations could then be weeded out and placed in a reserve set where they can be called up if necessary. This selection for membership would have to be done rather tediously by hand, in the current ERIL architecture. For a future architecture it may be desirable to have membership conditions on sets and introduce commands such as

```
m A next R | S | T
```

and

```
m A next R  S  T
```

The first command would try to put the next A equation into R or S or T, placing it into the first set whose membership condition it satisfies . The second command would try to place the equation in all three sets, if it satisfied their respective membership conditions.

The second, fair strategy which suggests itself, is to select equations on a probability basis. We set a probability for selecting equations between the equation sets. If the set probability was 1:10 say between sets A and H, there would be a one in ten chance of selecting an equation from A to move into T (and/or R), compared to a 9:10 chance of choosing an equation in H to move to Q. This would mean that in the limiting case there would be a 100% chance of an equation being selected, if combined with a fair strategy of choosing an equation from A or H themselves. Also there would be a bias towards generating subgoals from the original goal, and so the strategy would be more goal directed than the normal refutational proof methods. Of course, a probability is a non-deterministic, non-computable criteria to use. Consequently, in order to implement this we would have to pick every tenth equation from A, or use such devices as a psuedo-random number generator to pick the next equation.

5. Unfailing Knuth-Bendix Completion

The technique of Unfailing Knuth-Bendix Completion [HsiRus86] is an extension of the standard Knuth-Bendix Completion technique to enable it to handle unorientable critical pairs. It provides a decision procedure for many equational theories, and Hsiang and Rusinowitch give the decision procedure on an equational hypothesis as the main use of the procedure.

The problem which Hsiang and Rusinowitch identify and seek to address is that of the Knuth-Bendix Procedure failing due to the generation of rules which are unorientable under the term ordering chosen for the particular term algebra. The normal response of the Knuth-Bendix completion would be to fail and stop. The user would then have to try another ordering, which may not always be possible. Hsiang and Rusinowitch try another approach.

5.1. The Method

In the unfailing Knuth-Bendix procedure, the discovery of an unorientable equation does not cause the completion procedure to fail. The equality is kept as an equation and used to participate further in the procedure. To maintain the termination condition for rewriting, we insist that the ordering is a *Strong Simplification Ordering*. That is not only must it satisfy the normal properties for a simplification ordering (see [Ders87]), it must also be total upon ground terms. Thus although the commutative equation $x*y = y*x$ is not be orientable for variables x and y, a given ground term $a*b$ can be reduced to $b*a$ provided that $a*b >_t b*a$. They claim that most of the orderings normally used can be modified to make them into strong simplification orderings in a straightforward manner.

Hsiang and Rusinowitch redefine the notion of a Critical Pair to include equations as well as rewrite rules.

Definition: Given two equations $l_1 = r_1, l_2 = r_2$ such that for some substitution σ, $\sigma(l_1/u) = \sigma(l_2)$, that is the second equation superposes on the first at some subterm at the occurrence u, and

(i) $\sigma(r_1) \not\geq \sigma(l_1)$

(ii) $\sigma(r_2) \not\geq \sigma(l_2)$

with respect to some strong simplification ordering $>_t$ (with $\geq$ meaning either $>_t$ or $=_E$) then the *extended critical pair* is defined to be the pair of terms $(\sigma(l_1[u \leftarrow \sigma(r_2)]), \sigma(r_1))$. The extended critical pairs of a set of equations U are denoted $ECP(U)$.

They also allow terms to be reduced by equations. This is made more precise in the following definition.

Definition: A term s is reducible by an equation $l = r$ by a substitution σ which satisfies the following conditions.

(i) $\sigma(l) = s/u$ for some occurrence u

(ii) $\sigma(l) >_t \sigma(r)$ with respect to the strong simplification ordering $>_t$.

Then s is reduced to the term $s[u \leftarrow \sigma(r)]$. This is known as an *unfailing reduction*, and is denoted $\rightarrow^U$. Note that we only reduce by equations under substitutions which do orient the equation. Thus the rewriting is a true reduction in terms of the term ordering.

If the equations are orientable then these definitions correspond exactly to the normal Critical Pairs and reduction definitions. Thus rather than keep all the equalities as unoriented equations and apply the above critical pair and reductions rules, we can orient some rules into rewrite rules. We place the unorientable axioms another set in equational form and generate new critical pairs by superposing between the two sets as well as within the set of rewrite rules. The extended critical pairs of a set of equations U and a set of rewrites R are denoted $ECP(U \cup R)$. Reduction by these sets is denoted by $\rightarrow_R^U$.

Hsiang and Rusinowitch then present their algorithm as a series of informal inference rules. Bachmair [Bach87] gives a more formal account of the inference rules, similar to those I give below. He recognises that the equation set can be split into two sets of orientable and unorientable equations at the outset, and incorporates the distinction in the rules. Thus U is the current set of equations, R a set of rewrite rules.

UKB1 **Orienting an Equation.** An Equation can be oriented into a rewrite rule if its terms stand in the term-ordering.

$$\frac{(U \cup \{ s = t \}, R)}{(U, R \cup \{ s \rightarrow t \})} \quad \text{if } s >_t t$$

UKB2 **Adding a new Equational Consequence.** New equations can be generated by finding critical pairs between the Rewrite-Rules and the set of equations.

$$\frac{(U, R)}{(U \cup \{ s = t \}, R)} \quad \text{if } (s,t) \in CP(R) \cup ECP(U \cup R)$$

UKB3 **Simplifying an Equation.** Equations can be simplified by applying the Rewrite Rules to their terms.

$$\frac{(U \cup \{ s = t \}, R)}{(U \cup \{ u = t \}, R)} \quad \text{if } s \rightarrow_R^U u$$

UKB4 **Deleting a Trivial Equation.** Equations with identical terms are irrelevant and can be deleted.

$$\frac{(U \cup \{ s = s \}, R)}{(U, R)}$$

These rules are much the same as those for the normal Knuth-Bendix Completion. There are also a pair of simplification rules similar to those of the standard Knuth-Bendix Completion, except that rewriting is $\rightarrow_R^U$. Only the second rule, that of generating new equational consequences by the extended critical pair criteria differs by any significant extent. The procedure says, if a critical pair is orientable then by all means orient it, but do not worry if you cannot. Critical pairs are generated by the extended critical pair criteria, and so the equation set participates in the generation of new equations. This makes the process more inefficient but means that the procedure will not fail due to unorientable equations being formed.

The significance of the Unfailing Knuth-Bendix Procedure to Theorem Proving is that, as Bachmair proves, if the ordering on terms used is a strong simplification ordering, then a fair application of the Unfailing Knuth-Bendix Procedure will be canonical on ground terms. That is the procedure if it converges will produce a rewriting system which will reduce ground terms to unique normal forms. Of course like normal Knuth-Bendix completion, the process may fail to terminate, and then we have only a semi-decision procedure on ground terms.

By being limited to ground terms, this method may seem of limited application. However Hsiang and Rusinowitch provide a refutational approach to theorem proving which is more general. To prove a theorem $s = t$ in the term algebra T under the equational system (E,R) they first negate, and then Skolemise the formula to $\hat{s} \neq \hat{t}$, where $\hat{s}$ is the the term resulting from replacing existentially quantified variables by Skolem constants. As the original equality was implicitly universally quantified, in negating the formula, it becomes existentially quantified in all its variables, and so in Skolemising all variables are turned into Skolem constants, and thus the terms are converted into ground terms. Thus they can now be reduced to unique normal forms by the Unfailing Knuth-Bendix Procedure.

This is actually carried out by adding the equation $EQ(\hat{s},\hat{t}) = FALSE$ to E together with

the additional axiom $EQ(x,x) = TRUE$, running the procedure in the hope of eventually producing the contradiction $TRUE = FALSE$. Then the original equation must hold true. This strategy is proved refutationally complete both by Hsiang and Rusinowitch, and by Bachmair.

5.2. An Example

Bachmair Bach87 gives an example (which he in turn attributes to Hsiang) of the use of the unfailing procedure for the proof a simple theorem in a simple algebraic system. We start with the axioms.

$$\begin{array}{ll} x+y = x+x & \text{E1} \\ (x-y)+z = (x+z)-y & \text{E2} \\ (x+y)-y = x & \text{E3} \end{array}$$

If a Recursive Path Ordering is used, in which we can choose "–" to be declared of lower precedence than "+", we can orient E2 and E3 into the rewrite rules below, using inference rule UKB1. We have to use a total ordering on ground terms, and so we have to declare the relative values of all constants and operators in advance, including any Skolem constants used.

$$\begin{array}{ll} (x-y)+z \rightarrow (x+z)-y & \text{R1} \\ (x+y)-y \rightarrow x & \text{R2} \end{array}$$

However E1 cannot be oriented sensibly, so we leave it unoriented, but since we are applying Unfailing Knuth-Bendix Completion, we can still use it to provide extended critical pairs.

We wish to prove that under this algebra, $(x-y)+z = x$. We add the Skolemised refutation, $EQ((a-b)+c,a) = false$ to a hypothesis set, together with the reflexivity axiom for the equality predicate, $EQ(x,x) = true$. The hypothesis rewrites immediately by R1 to form the new hypothesis

$$EQ((a+c)-b,a) = false$$

Rule R1 and the remaining equation E1 superpose to form the critical pair, by UKB2.

$$(x+z)-y = (x-y)+(x-y)$$

which simplifies by R1, by UKB3 to

$$(x+z)-y = (x+(x-y))-y \qquad \text{E4}$$

This rule is not orientable, but will superpose with R2 to form an extended critical pair by UKB2, which can be oriented by UKB1 to form the rule

$$(x+(x-y))-y \rightarrow x \qquad \text{R3}$$

By UKB3, this simplifies the equation E4 to a form where it can be oriented into a rule

$$(x+z)-y \rightarrow x \qquad \text{R4}$$

This rule rewrites the goal to the form $EQ(a,a) = false$, which in turn rewrites by the reflexivity of EQ to form the contradictory equation $T = F$.

This example illustrates the use of the extended critical pair mechanism, although we do not actually use the totality of the ordering in the proof. This is not necessarily the best way of proceeding towards the refutation. It uses a rewriting method rather like the constructive proof methods, without any interaction between the goal and the equalities. The process could have put the goal in the axiom set where it would have formed (extended) critical pairs. Neither is it the way an automated implementation would proceed. We shall see later how ERIL actually does perform on this example.

5.3. Configurations for Unfailing Completion: A Simple Theorem Proving Example

Dick and Kalmus [DicKal88] give a brief account of how to use the ERIL system for unfailing completion, but do not give an example of using this system for Theorem Proving. To show that it can be used in this way, the example given by Bachmair [Bach87] and described earlier is given.

We set up a configuration of the ERIL system as follows. All reduction links apart from those reducing the hypotheses are excluded for the sake of clarity.

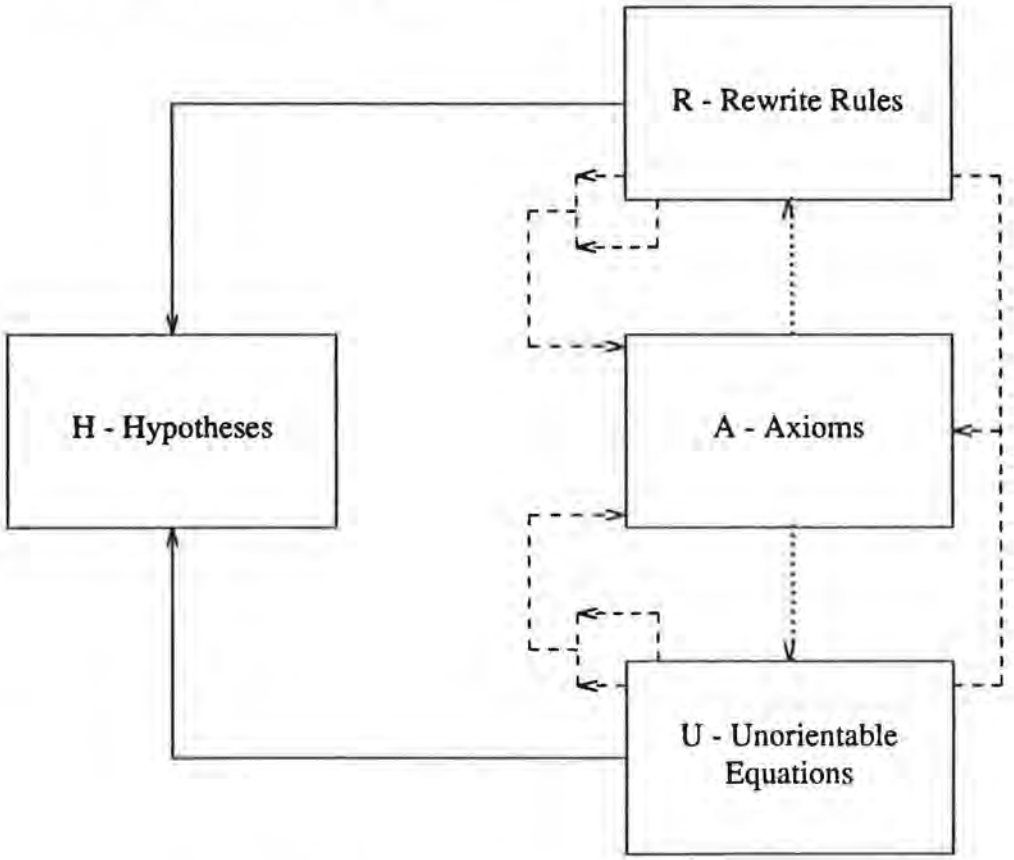


Figure 10: A Configuration for Unfailing Completion.

Here there are four equality sets. Rewrite Rule and Axiom sets are set up as usual. The third set U holds those axioms which cannot be oriented to be entered into R. Critical pairs can either be generated in the normal way, or else be created as extended critical

pairs by superposing either side of an orientable axiom in *U*, with either a rule in *R* or a equation in *U*. All critical pairs are placed into *A* for further consideration. There are reduction links for *R* to reduce *R,A,U,H* by rewriting, and for *U* to reduce *R,A,U,H*, but only in cases where the instance is orientable with respect to the termination ordering.

This is an implementation of the rules previously given. The set *E* there which contains all the current equations, whether orientable or not, has been here split into two. The axiom set *A* contains equations which have yet to be considered for ordering. If they are orientable, they are formed into rules. Only equations which have been proven to be unorientable are entered into *U*. Thus we keep the unorientable equations down to a minimum.

For our example we enter the following sets of equations. For this example a global Recursive Path Ordering is defined in advance to ensure a total ordering on ground terms.

```
ERIL version R1.6a        Rutherford Appleton Laboratory - Informatics Division
------------------- Unfailing KNUTH-BENDIX Configuration ---------------------
R     Rewrite rules                0
-------------------------------------------------------------------------------
A     Axioms                       4
A1:  x+x1 = x+x
A2:  EQ(x,x)
A3:  (x-x2)+x1 = (x+x1)-x2
A4:  (x+x1)-x1 = x
-------------------------------------------------------------------------------
U     Unoriented Axioms            0
-------------------------------------------------------------------------------
H     Hypotheses                   1
H1:   EQ(((a-b)+c),a) =?= F
    { EQ(((a-b)+c),a) =?= F }
-------------------------------------------------------------------------------
Press ENTER/RETURN key to continue :                                (f/Finish)
-
```

A1 is unorientable and so is put into *U*. The important step in this proof is the extended critical pair `x = (x+x)-x1` formed between A1, renamed as U1 and the rule resulting from A4. This, when made into a rule, reduces the goal, which has already been reduced by other rules including an orientable instance of U1, to `EQ(((a+a)-b),a) =?= F`. The result is `EQ(a,a) =?= F`, which of course creates a contradiction with A2. However, the process is not as neat as that, nor as short as the derivation given previously. The final situation is as follows.

```
ERIL version R1.6a          Rutherford Appleton Laboratory - Informatics Division
-------------------- Unfailing KNUTH-BENDIX Configuration ---------------------
R     Rewrite rules                4
R2:  EQ(x,x)
R3:  (x-x2)+x1 => (x+x1)-x2
R4:  (x+x1)-x1 => x
R6:  (x+x)-x1 => x
--------------------------------------------------------------------------------
A     Axioms                       11
A22: x-x2 = x-x1
A11: x = (x+x1)-x
A14: (x+x3)-x2 = (x+x1)-x2
A21: x = (x+x1)-x2
A8:  x-x2 = ((x+x1)-x2)-x1
A9:  ((x+x2)+x1)-x2 = x+x1
A17: ((x+x)+x2)-x3 = ((x+x1)+x2)-x3
A20: ((x+x3)+x1)-x2 = ((x+x)+x1)-x2
A19: x-x3 = ((x+x1)-x2)-x3
A16: ((x+x2)-x3)-x1 = x-x1
A23: ((x+x)+x1)-x2 = x+x1
--------------------------------------------------------------------------------
U     Unoriented Axioms            1
U1:  x+x1 == x+x
--------------------------------------------------------------------------------
H     Hypotheses                   1
H1:   EQ(((a-b)+c),a) =?= F
    { T =?= F }
--------------------------------------------------------------------------------
Press ENTER/RETURN key to continue :                                (f/Finish)
-
```

The process creates other extended critical pairs unnecessary for the theorem proving process. Eighteen extended critical pairs are created in all, while in actual case we needed only one, the rest of the problem being solved by reduction by rules resulting from the initial axiom set.

The above configuration is thus adequate for basic unfailing theorem proving, but far from ideal. Note that the hypothesis did not play a guiding role in discovering the path to the solution, and consequently extraneous critical pairs were generated. The method is very similar to the constructive methods of theorem proving demonstrated in [Dick-Kal88]. In general the goal needs to be in Skolemised, refutational form because the global ordering on ground terms meant that the unorientable equations could be applied to it, and not for reasons of strategy. There is scope for experimenting with strategies for unfailing completion theorem proving in a similar manner to the above configurations for Paul's method, in order to make the process more goal directed.

6. Conclusions

The ERIL system proves a successful vehicle for experimenting with different strategies. What the work demonstrates is that while on the whole certain strategies often perform better than others, there are special cases where the "better" strategy performs poorly, and the "worse" well. Therefore it would be advantageous to have variety of strategies for equational reasoning tasks available, and the approach of the ERIL system caters for

that flexibility. By changing the selection strategy significant improvements in performance can (sometimes) be achieved. This often requires some ingenuity, but by and large, once carefully set up, a selection strategy could be left to solve a problem on its own. However, the current ERIL system is not without its problems and difficulties.

6.1. Problems with the ERIL system

The configuring of ERIL for the above work has revealed several problems with the existing system. These made the work more complicated than necessary, and meant that the strategies implemented were not as efficient as they could have been.

1. **Link Attributes**
   As pointed out in [Matt88], there is a problem with the way that the `link` attribute in ERIL is implemented. This attribute determines which set an equality should be placed into after it has been modified by rewriting or subsumption, or generated by superposition. In the ERIL system, this is a global attribute for each set, no matter how the modification arose. One consequence of this is that if a rewrite rule is rewritten on either side, it is placed in the same set. By inference rule. KB6 above, we want to place it back into the set of rules if it is rewritten on the right, as it is oriented the correct way and will not generate any critical pairs which have not already been considered. However, by KB5, if it is rewritten by on the left it has to be placed into the set of equations for further consideration as a rule rewriting (if it can be oriented) and also for generating critical pairs. This separation cannot be performed in ERIL, and so KB6 has not been used in the above strategies. This means that equations are being reconsidered for critical pairs unnecessarily.

2 **Failure to Delete an Active Rule**
   In the simplifying strategies, it can occur that a child critical pair is oriented and rewrite one of its parent rules. If that parent is the one being entered into the rule set, then the critical pair generation using that rule should stop. The modified rule (if it has not been deleted) will receive consideration again later in the process. However, the ERIL system will continue the process of generating the critical pairs of the unmodified parent, and only remove it from the rule set at the end of the superposition process. Thus unnecessary critical pairs are generated.

3 **Automating Commands**
   It is difficult to fully automate the completion strategy, particularly in the strategies which have many different rule sets. This means that the user has to constantly intervene to repeat the command line to drive the configuration. This does not make the completion less automatic: once the configuration has been set the user makes no decisions as to the strategy taken by the inference engine. But it does make the completion task very tedious. The flexibility of ERIL is very useful, but it does not go far enough. It would be desirable to be able to set an automatic repeat on the command line.

4 **Proving Configurations Correct.**
   Demonstrations of the "correctness" of the given configurations for equational completion have been given. However, these arguments seem informal and unconvincing. Correspondences have been informally drawn between sections of the ERIL configuration and control and the inference rules of completion. This does seem a most unsatisfactory form of proof. It would be desirable to have available some sort

of tactic language in which the user could design configurations to perform various equational reasoning tasks in a flexible manner, like the ERIL system, but at the same time be manipulating objects with a formal semantics so that correctness proofs of the resulting algorithms can be produced.

6.2. Future Work.

The author is engaged in a new implementation of the ERIL system which will extend the current ERIL system with AC operators, and also incorporate many of the ideas given in this paper, particularly with respect to a tactic language. It is desired that the user will be able to set up his or her configuration using a tactic language, with a direct correspondence of links in the ERIL sense and inference rules, and that from the formal semantics of that language, carry out proofs of correctness of the implementations. Thus we combine the flexibility of the ERIL configuration mechanism with the security of proof.

Acknowledgements

Thanks to Juan Bicarregui, Stuart Robinson and others in Systems Engineering at RAL, Jeremy Dick who developed the ERIL system, and Muffy Thomas, my supervisor at Glasgow University.

References

[Bach87]. Leo Bachmair, "Proof Methods for Equational Theories," PhD Thesis, University of Illinois (1987).

[Bach91]. Leo Bachmair, *Canonical Equational Proofs*, Birkhauser (1991).

[Ders87]. Nachum Dershowitz, "Termination of Rewriting," *J. of Symbolic Computation* **3** pp. 69-116 (1987).

[DerJou90]. Nachum Dershowitz and Jean-Pierre Jouannaud, "Rewrite Systems.," pp. 243-320 in *Handbook of Theoretical Computer Science,*, ed. J. van Leeuwen, Elsevier Science Publishers (1990).

[Dick87]. A J J Dick, "Order-Sorted Equational Reasoning and Rewrite Systems," PhD Thesis, Imperial College, University of London (1987).

[DicKalMar88]. A J J Dick, J R Kalmus, and U H Martin, "Automating the Knuth-Bendix Ordering," *submitted to Acta Informatica*, (1988).

[DicKal88]. A J J Dick and J R Kalmus, "ERIL (Equational Reasoning: an Interactive Laboratory) User's Manual," Rutherford Appleton Laboratory Report, RAL-88-055. (1988).

[Dick91]. A J J Dick, "An Introduction to Knuth-Bendix Completion.," *The Computer Journal* **34**(1) pp. 2-15 (1991). Also Rutherford Appleton Laboratory Report, RAL-88-043

[Forg84]. Randy Forgaard, "A Program for Generating and Analyzing Term Rewriting Systems," MSc Dissertation, Massachusetts Institute of Technology (1984).

[GarGut89]. Stephen J Garland and John V Guttag, "An Overview of LP, the Larch Prover.," pp. 137-151 in *Proc. 3rd Conference on Rewriting Techniques and Applications, Chapel Hill, NC, USA. Lecture Notes in Computer Science, no. 355.*, ed. N Dershowitz, Springer-Verlag (1989).

[HsiDer83]. Jieh Hsiang and Nachum Dershowitz, "Rewrite Methods for Clausal and Non-Clausal Theorem Proving," pp. 331-346 in *Lecture Notes in Computer Science*, ed. Diaz J , Springer-Verlag (1983).

[HsiRus86]. Jieh Hsiang and Michael Rusinowitch, "On Word Problems in Equational Theories," Draft Report, Dept of Computer Science, State University of New York (1986).

[HuOp80]. Gerard Huet and Derek C Oppen, "Equations and Rewrite Rules: A Survey," pp. 349-393 in *Formal Language Theory: Perspectives and Open Problems*, ed. Ronald V Book, Academic Press (1980).

[Huet81]. G. Huet, "A Complete Proof of the Correctness of the Knuth-Bendix Completion Algorithm," *Journal of Computer and System Science* **23**(1) pp. 11-21 (1981).

[Kuch83]. Wolfgang Küchlin, "A Theorem Proving Approach to the Knuth-Bendix Completion Algorithm," pp. 101 - 108 in *Lecture Notes in Computer Science 144, EUROCAM Computer Algebra Conference*, ed. Calmet J , Springer-Verlag (1983).

[KapZha89a]. Deepak Kapur and Hantao Zhang, "An Overview of the Rewrite Rule Laboratory (RRL)," pp. 559-563 in *Proc. 3rd Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science 355*, ed. N Dershowitz, Springer-Verlag (1989).

[KnuBen70]. Donald E. Knuth and Peter B. Bendix, "Simple Word Problems in Universal Algebras," pp. 263 - 297 in *Computational Problems in Abstract Algebra*, ed. Leech J, Pergamon Press (1970).

[Lesc89]. Pierre Lescanne, "Completion Procedures as Transition Rules + Control," Centre de Recherche en Informatique de Nancy, Internal Report. (1989).

[Lesc90]. Pierre Lescanne, "Implementation of Completion Procedures as Transition Rules + Control: ORME," pp. 262-269 in *Proc. of 10th ACM Symp on Principles of Programming Languages. Lecture Notes in Computer Science, No. 463.*, Springer-Verlag (1990).

[Love78]. Donald W Loveland, *Automated Theorem Proving: A Logical Basis*, North-Holland (1979).

[Matt88]. Brian M Matthews, "Strategies for Theorem Proving in an Equational Reasoning System," MSc Dissertation, Imperial College, University of London (1988).

[Paul84]. Etienne Paul, "A New Interpretation of the Resolution Principle," in *Proc. of CADE 7, Napa California,* , Springer-Verlag ().

[Paul85b]. Etienne Paul, "On Solving the Equality Problem in Theories Defined by Horn Clauses," in *Proc. of EUROCAL 85, Linz Austria, Lecture Notes in Computer Science 203.*, Springer-Verlag (1985).