DEDUCTION PLANS: A BASIS FOR INTELLIGENT BACKTRACKING

P.T. Cox
T. Pietrzykowski

Research Report CS-79-41

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

December 1979

Deduction Plans

P.T. Cox
T. Pietrzykowski

University of Waterloo, Waterloo, Ontario, Canada

ABSTRACT: A proof procedure is described that relies on the construction of certain directed graphs called "deduction plans". Plans represent the structure of proofs in such a way that problem-reduction may be used without imposing any ordering on the solution of subproblems, as required by other systems. The structure also allows access to all clauses deduced in the course of a proof, which may then be used as lemmas. Economy of representation is the maximum attainable, consistent with this unrestricted availability of lemmas.

Various restrictions of this deduction system are seen to correspond to existing linear deduction procedures, while overcoming many of their shortcomings. One of the rules for constructing plans, however, has no equivalent in existing systems.

This research was supported by Natural Science and Engineering Research Council Grants: A5267 and A3025.

Authors' address: Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1.

A further economy is obtained by obviating the necessity for explicitly performing substitutions and for calculating most general unifiers.

The source of unification failure can be located when a subproblem is found to be unsolveable, so that exact backtracking can be performed rather than the blind backtracking performed by existing systems. Therefore, a deduction system based on the construction of plans can avoid the wasteful search of irrelevant areas of the search space that results from the usual backtracking methods. Furthermore, because of the graphical structure, it is necessary to remove only the offending parts of the proof when a plan is pruned after backtracking, rather than the entire proof constructed after the cutting point.

KEY WORDS AND PHRASES: Resolution, theorem-proving, first order logic, directed graph, unification, linear deduction, backtracking, plan.

CR CATEGORIES: 3.6, 5.21

1. Introduction

Early theorem-proving programs based on Robinson's resolution principle [14], employed the "saturation" search strategy, in which the search space is exhaustively generated level by level. For obvious reasons, the performance of such programs was disappointing. Many search strategies were proposed in the late sixties for improving the performance of resolution-based provers. One of these, "linear resolution", independently proposed by Loveland [10], Luckham [12], and Zamov and Sharonov [15], employed the well-known technique of "problem-reduction", and appeared to be one of the more promising of these refinements.

Existing linear deduction systems suffer from several drawbacks. First, to ensure completeness in the original linear resolution system, it is necessary for any clause deduced in the course of a proof to be accessible as a "lemma": that is, it must be available for use as though it were a member of the set of clauses whose unsatisfiability is in question.

This of course requires that copies are kept of all clauses deduced, and that this list is continually scanned for useful lemmas. The storage problem could be overcome by the use of a structure-sharing scheme such as that proposed by Boyer and Moore [4], however, the list of lemmas must still be scanned. A solution to this problem is to use the linear system proposed by Loveland as Model Elimination [8,9,11], and Kowalski and Kuehner as SL-resolution [7]. In these systems, a rule is used which corresponds to the familiar proof technique "reductio ad absurdum". To ensure soundness, however, a strict ordering must be imposed on the solution of subproblems. Consequently, we lose one of the most attractive features of problem reduction,

the parallel processing of subproblems.

Linear deduction also suffers from two problems associated with "backtracking". If a subproblem is found to be unsolveable in the course of a proof, the system must return to an earlier state of the proof and attempt an alternative solution to a previously solved subproblem. The strategy normally adopted by a linear theorem-prover is to return to the last subproblem it solved for which there is an untried potential solution: however this may not be the correct place to try an alternative, and although the correct point will be reached eventually, much effort may be expended in the meantime on a fruitless exploration of the search space. This inefficiency is compounded by the pruning that occurs whenever the system backtracks: when the linear prover returns to a subproblem A to attempt an alternative solution, it erases all the clauses produced since the last attempt at solving A. Some of these clauses may constitute a perfectly acceptable solution to some subproblem, and may eventually be regenerated. Furthermore, because of its blind backtracking behaviour, the linear system may backtrack over this innocuous subproof several times, and regenerate it several times.

We present here a deduction system which has all the advantages of the linear systems, but overcomes their above-mentioned deficiencies.

A proof is represented as a directed graph, called a "deduction plan" the vertices of which are occurrences of literals from the set of clauses

under consideration. In building plans, the process of constructing a proof is separated from the unification process. The word "plan" is intended to convey the notion that such a graph proves nothing until the rules used in its construction have been validated by a unification algorithm. In this respect, our system is similar to Huet's higher-order "constrained resolution" [6].

To each rule of the various linear deduction schemes, there corresponds a rule for plan construction; however, even though the rules for Model Elimination have equivalents in plan construction, no ordering need be imposed on the solution of the subproblems of a plan to obtain soundness. One rule, called "backfactoring", has no equivalent in linear deduction, and involves factoring a subproblem to a previously solved subproblem. It is seen that completeness is preserved if all factorings are restricted to be of this type.

Plans allow the use of lemmas as in simple linear deduction, but more lemmas are available. This is because each plan actually corresponds to a set of linear deductions, and any clause deduced in any one of these deductions is available as a lemma. Because of the graphical structure of plans, however, the storage problem caused by keeping lemmas in linear deduction is eliminated in our system. In fact, among systems which allow complete access to the history of a proof, plans are the most economical, since a plan contains only one representation of each literal used in a proof.

A further economy is attained in that no substitutions need ever be performed: it is necessary only to verify that certain expressions are unifiable, and therefore, most general unifiers need not be calculated.

Although we do not discuss it in this paper, there exists an algorithm

for determining the source of unification failure, and hence for finding all choices for backtracking [5]. Once this has been done the proof must be pruned, and the graphical structure ensures that no harmless subproofs are removed. The algorithm for tracing unification failure is the subject of a forthcoming publication.

In section 2, we present some definitions, in section 3 define plans, and in section 4 we prove the soundness and completeness of deduction systems based on them. In section 5 we discuss practical aspects of plans.

2. Preliminaries

Here we provide some notation, make preliminary definitions, and quote familiar results for later reference.

2.1: Graph Theory

With a few minor exceptions, our notation and definitions for the concepts of graph theory follows Bondy and Murty [3].

2.1.1: Definition: A directed graph G is an ordered pair $< V(G), E(G) >$ where $V(G)$ is a nonempty set of vertices and $E(G) \subseteq V(G) \times V(G)$ is the set of arcs. If $e = (u,v)$ is an arc, then e is said to join u to v, u is called the tail of e, and v is called the head of e. We also say that e leaves u and enters v. The indegree and outdegree of a vertex v, are respectively the number of arcs which enter v, and the number of arcs which leave v.

We will henceforth abbreviate "directed graph" to "digraph".

2.1.2: Definition: A digraph D is a subdigraph of a digraph G if $V(D) \subseteq V(G)$ and $E(D) \subseteq E(G)$.

2.1.3: Definition: Let V be a set and $E \subseteq V \times V$. We shall say a graph G is induced by E iff

$$E(G) = E$$

and

$V(G) = \{ x \in V | \exists e \in E$ such that x is the head or the tail of $e\}$. We shall denote G as $IND(E)$.

2.1.4: Definition: If G is a digraph, a directed walk in G is a sequence of arcs $e_1,\ldots,e_n$ $(n \geq 1)$ such that the tail of e_{i+1} is the head of e_i for $1 \leq i < n$. Note that this walk can also be specified by the sequence $v_1,\ldots,v_{n+1}$ of vertices, where $e_i = (v_i,v_{i+1})$ for $1 \leq i < n$. The length of the walk is n; v_1 and v_{n+1} are respectively the origin and terminus, and $v_2,\ldots,v_n$ are called the internal vertices of the walk. A directed walk is said to be closed if its origin and terminus are identical.

We will frequently use such expressions as "v lies on the walk"; "a walk from u to v"; and "the walk passes through v". The meaning of such expressions is obvious. Since we consider only directed graphs, we will usually omit the word "directed" and the prefix "di-", using "graph", "walk", "subgraph", etc., instead of "directed graph", "directed walk", "subdigraph", etc.

2.2: Language:

In what follows, we will use the expressions "variable", "constant", "predicate symbol", "function symbol", "term", "atom" and "well-formed expression" with their usual meaning (see Robinson [14]). Our definitions of "literal" and "clause" are somewhat different, however.

2.2.1: Definition: A literal is either an atom or a string of the form -A where A is an atom. If L is a literal, the negation of L, denoted ¬L, is the literal:

(i) -A if L is the literal A, where A is an atom

(ii) A if L is the literal -A.

A ground literal is a literal in which no variable occurs.

2.3: Substitution and Unification:

The notions of "substitution", "application of substitution", and "composition of substitution" follow Robinson [14] with the exception that a substitution component will be denoted (v,t) rather than t/v, where v is called a replaced variable. From now on we will refer to "well-formed expressions" simply as "expressions".

2.3.1: Definition: A substitution θ unifies a set of expressions E if and only if $E\theta$ contains one element. In this case, E is said to be unifiable, and θ is a unifier for E. θ is called a most general unifier (mgu) for E if and only if for every unifier γ for E, there is a substitution β such that $\gamma = \theta \cdot \beta$.

We extend the notion of unifiability to sets of sets of expressions.

2.3.2: Definition: If $\mathcal{E}$ is a set of sets of expressions and θ is a substitution, θ unifies $\mathcal{E}$ if and only if θ unifies E for each $E \in \mathcal{E}$. We define "unifiable", "unifier", and "most general unifier" exactly as in 2.3.1.

The fact that every unifiable set has at least one mgu is clear from the existence of several unification algorithms.

If E is a unifiable set of expressions we denote by mguE, some mgu of E. We use this notation also for sets of sets of expressions. Note that if E is a set of expressions, then $\text{mguE} = \text{mgu}\{E\}$.

Although the following result is fairly obvious, we provide a proof since, to our knowledge, no proof has been published to date.

2.3.3: Lemma: If X_1 and X_2 are both sets of expressions, or both sets of sets of expressions, then:

(i) $X_1 \cup X_2$ is unifiable if and only if X_1 is unifiable and $X_2 \text{mgu} X_1$ is unifiable.

(ii) If $X_1 \cup X_2$ is unifiable:

$$\text{mgu}(X_1 \cup X_2) = \text{mgu}X_1 \circ \text{mgu}(X_2 \text{mgu} X_1)$$

Proof:

(i) (a) If $X_1 \cup X_2$ is unifiable, let θ be any unifier for $X_1 \cup X_2$. θ unifies X_1, so that by the definition of mgu, $\theta = \text{mgu}X_1 \circ \beta$ for some substitution β. But θ unifies X_2, so that β unifies $X_2 \text{mgu} X_1$. Therefore X_1 and $X_2 \text{mgu} X_1$ are unifiable.

(b) Suppose X_1 and $X_2 \text{mgu} X_1$ are unifiable, and let θ be a unifier for $X_2 \text{mgu} X_1$. Now $\text{mgu}X_1$ unifies X_1, so that $\text{mgu}X_1 \circ \theta$ unifies X_1; also θ unifies $X_2 \text{mgu} X_1$ so that $\text{mgu}X_1 \circ \theta$ unifies X_2. Hence $\text{mgu}X_1 \circ \theta$ unifies $X_1 \cup X_2$ so that $X_1 \cup X_2$ is unifiable.

(ii) If $X_1 \cup X_2$ is unifiable, then by part (i), $mgu(X_2 mguX_1)$ exists. Now $mguX_1$ unifies X_1, so that $mguX_1 \circ mgu(X_2 mguX_1)$ unifies X_1. Also:

$$X_2 mguX_1 \circ mgu(X_2 mguX_1) = (X_2 mguX_1) mgu(X_2 mguX_1)$$

So $mguX_1 \circ mgu(X_2 mguX_1)$ clearly unifies X_2. Therefore $mguX_1 \circ mgu(X_2 mguX_1)$ is a unifier for $X_1 \cup X_2$.

Suppose θ is a unifier for $X_1 \cup X_2$, then since θ unifies X_1:

$$\theta = mguX_1 \circ \beta \quad \text{for some substitution } \beta$$

But θ unifies X_2, so that β unifies $X_2 mguX_1$.

$$\therefore \quad \beta = mgu(X_2 mguX_1) \circ \alpha \quad \text{for some substitution } \alpha$$

$$\therefore \quad \theta = mguX_1 \circ mgu(X_2 mguX_1) \circ \alpha$$

Therefore $mguX_1 \circ mgu(X_2 mguX_1)$ is an mgu for $X_1 \cup X_2$. □

2.4: Predicate Calculus

As we remarked earlier, our definition of "clause" differs from Robinson's. Robinson regards a clause as a set of literals, and since a set can have no repeated members, the clause corresponding to the propositional formula pvp is {p}. To avoid confusion between distinct vertices of a graph when these vertices correspond with the same literal, we need to distinguish between different occurrences of the same literal. We achieve this by attaching distinct indices to these occurrences.

2.4.1: Definition: Let I be any countable set, and L the set of all literals; then the set $L \times I$ is called the set of elements over the index set I. We will assume from now on that I is fixed, and refer to $L \times I$ simply as the set of elements. If (x,i) is an element, we define $|(x,i)| = x$, and refer to i as the index of (x,i). If (x,i) is an element and θ is a substitution, then the element $(x\theta,i)$, also denoted $(x,i)\theta$, is called an instance of (x,i).

2.4.2: Definition: A finite subset C of $L \times I$ is called a clause iff:

either C is empty

or $C = \{(x_1,i_1),\ldots,(x_m,i_m)\}$, $m \geq 1$

where $i_1,\ldots,i_m$ are distinct indices.

We denote the empty clause by $\square$.

If $C = \{(x_1,i_1),\ldots,(x_m,i_m)\}$ and $D = \{(x_1,j_1),\ldots,(x_m,j_m)\}$ are clauses, then C is said to be a copy of D. A ground clause is one in which no variable occurs. A pair of clauses are said to be separated iff they have no variables or indices in common. If $C = \{x_1,\ldots,x_m\}$ is a clause and θ is a substitution, then the clause $\{x_1\theta,\ldots,x_m\theta\}$, also denoted $\{x_1,\ldots,x_m\}\theta$ is called an instance of C.

2.4.3: Definition: A substitution $\{(v_1,u_1),\ldots,(v_n,u_n)\}$ is called a renaming if $u_1,\ldots,u_n$ are distinct variables, and:

$$\{u_1,\ldots,u_n\} \cap \{v_1,\ldots,v_n\} = \phi$$

If γ is a renaming, then γ^{-1} is the renaming $\{(u,v)|(v,u) \in \gamma\}$. Clearly, if γ is a renaming, then $\gamma_o\gamma = \gamma$, $(\gamma^{-1})^{-1} = \gamma$, and $\gamma_o\gamma^{-1} = \gamma^{-1}$. A clause C is a variant of a clause D iff C and D have no variables in common and $C = D\gamma$ where γ is some renaming.

2.4.4: Definition: A valuation is a function Σ from the set of all ground literals into the set $\{T,F\}$ such that:

$$\Sigma(L) = T \text{ iff } \Sigma(\neg L) = F$$

We extend the domain of every valuation Σ to the set of all clauses, as follows:

(i) If C is a ground clause:

$$\Sigma(C) = T \text{ iff } \Sigma(|L|) = T \text{ for some } L \in C$$

(ii) If C is a clause which is not a ground clause:

$$\Sigma(C) = T \text{ iff } \Sigma(C\theta) = T \text{ for all ground instances } C\theta \text{ of } C$$

2.4.5: Definition: A valuation Σ is said to satisfy a clause C if and only if $\Sigma(C) = T$. We also say that Σ is a model for C. Similarly, a valuation Σ is said to satisfy, or to be a model for a set of clauses S if and only if $\Sigma(C) = T$ for all $C \in S$. A set S of clauses is satisfiable iff it has no model, otherwise it is unsatisfiable.

3. Deduction Plans

3.1: Introduction

In this section we present a deduction system which relies on the construction of certain directed graphs called "deduction plans". Before defining these graphs formally, we give an informal description of their structure.

The underlying structure of a deduction plan for a set S of clauses is a rooted tree the root of which is a special vertex called TOP. Every vertex other than TOP is an element of a copy of a variant of a clause of S. This underlying rooted tree corresponds exactly to a linear resolution deduction from S in which the only inference rule is binary resolution. The rule used in building such a tree is called "replacement". The following example illustrates such a rooted tree and the corresponding linear resolution deduction.

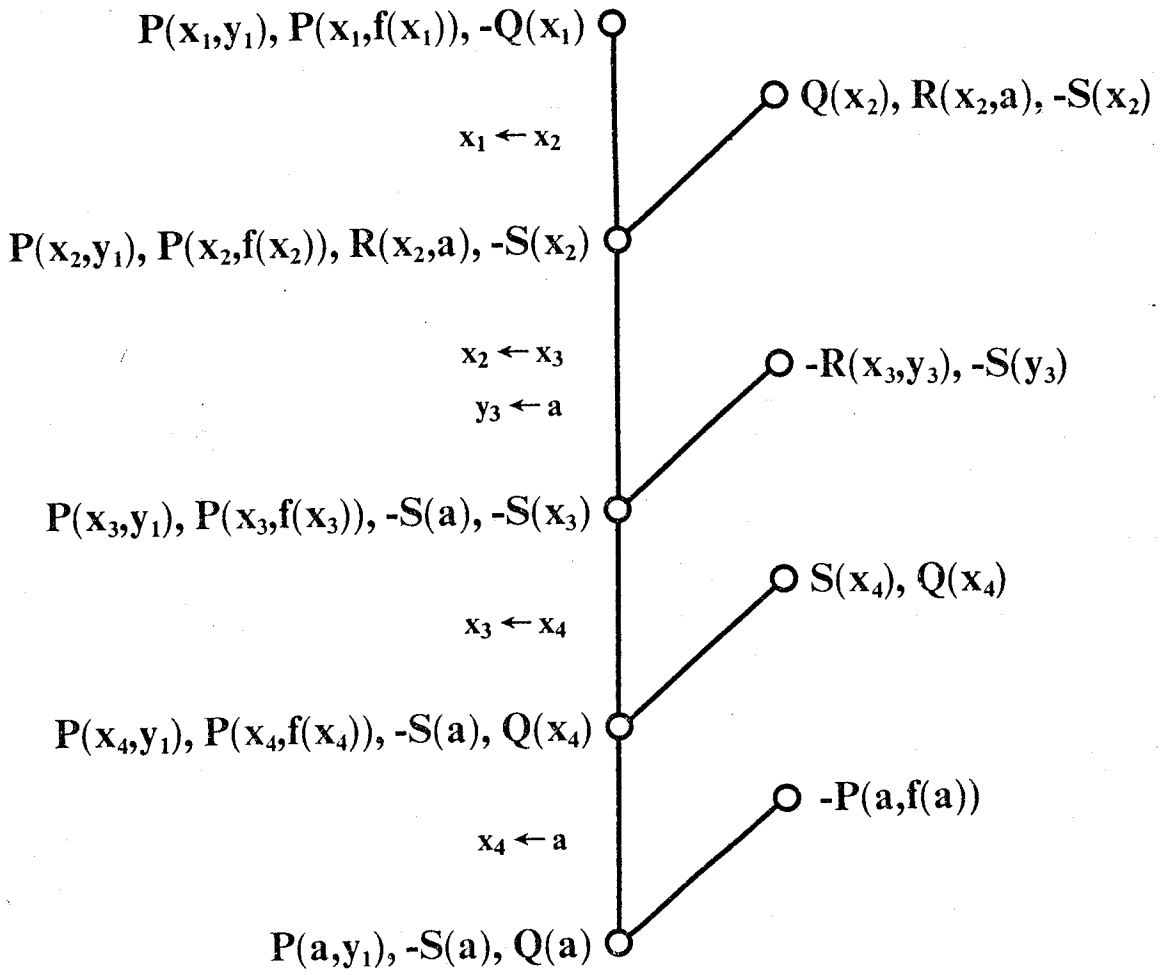
3.3.1: Example: Consider the set of clauses:

$$
\begin{aligned}
S = \{\ & \{P(x,y),\ P(x,f(x)),\ -Q(x)\},\\
& \{Q(x),\ R(x,a),\ -S(x)\},\\
& \{S(x),\ Q(x)\},\\
& \{-R(x,y),\ -S(y)\},\\
& \{-P(a,f(a))\} \qquad\qquad \}
\end{aligned}
$$

where a is a constant. In this and future examples we will omit indices from clauses, since in our pictorial representation of plans this will cause no ambiguity.

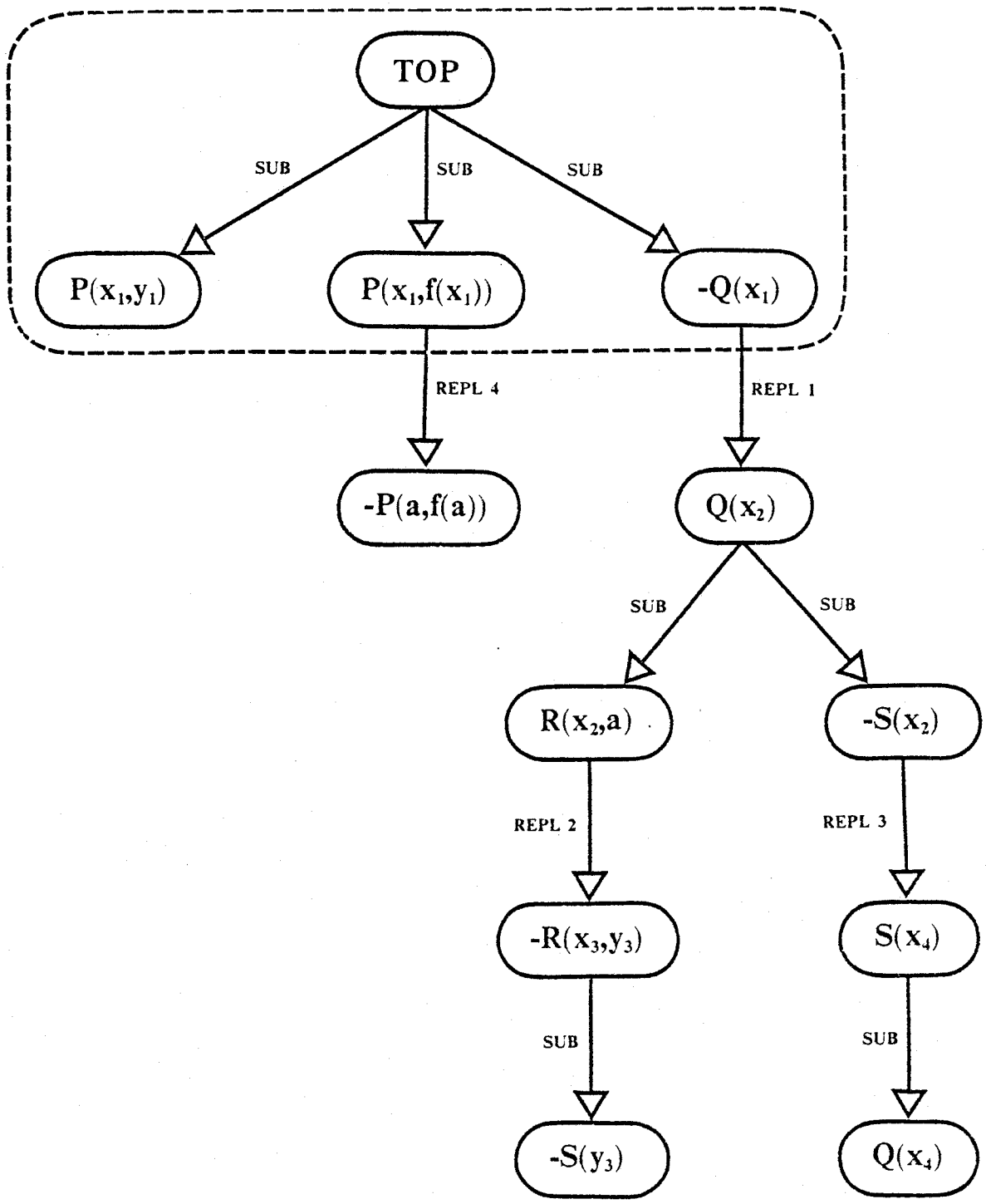
Figure 3.1 is a linear binary resolution deduction from S, and figure 3.2 illustrates the corresponding plan. Note that the subgraph indicated by the dotted line corresponds to the top clause in the linear deduction, and the arcs labelled "SUB" connect the root vertex with the elements of the top clause. SUB stands for "subproblem" and indicates elements which must be removed by resolution. If a subproblem has no arcs leaving it, then that subproblem is said to be "open": that is, it has yet to be removed by resolution. Each arc labelled "REPL" indicates one application of the replacement rule, and shows that we have selected a clause C which is a copy of a variant of some clause from S, and have performed a binary resolution on the subproblem at the tail of the arc, using the element of C which appears as the head of the arc. The remaining elements of C are then introduced as new subproblem vertices at the head of new SUB arcs: the tail of each of these new SUB arcs is the head of the new REPL arc. In figure 3.2 the REPL arcs are numbered, indicating the order in which the tree is constructed: this order corresponds exactly to the order in which deductions are performed in the linear resolution deduction of figure 3.1. When a subproblem becomes the tail of an arc it is said to be "closed". Note that a vertex at the head of a REPL arc is not a subproblem.

In building this tree we have not applied the unifying substitutions as in the linear deduction. Instead, a record is kept of the expressions which must be unified in order to validate the construction. In the case of the plan in figure 3.2, every pair of expressions in the set



A binary linear resolution deduction from the set of clauses of example 3.1.1. The substitution applied during each application of the resolution rule is noted beside the appropriate branch.

Figure 3.1



A plan for the set of clauses of example 3.1.1 corresponding to the linear resolution of figure 3.1.

Figure 3.2

$\{\{x_1,a\}, \{f(x_1),f(a)\}, \{x_1,x_2\}, \{x_2,x_3\}, \{a,y_3\}, \{x_2,x_4\}\}$ must be simultaneously unified. Note that the substitution $\{(x_1,a), (x_2,a), (x_3,a), (x_4,a), (y_3,a)\}$ is in fact a unifier for this set.

At each stage during the construction of this tree, the set of open subproblems corresponds to the clause deduced by the equivalent linear resolution. If at some stage there were no open subproblems left, then in the corresponding linear resolution, the empty clause would have been obtained, showing the set S to be unsatisfiable. The object, therefore, when using plans to prove unsatisfiability, is to construct a plan with no open subproblems: such a plan is said to be "closed".

Unfortunately, binary resolution is not complete: that is, for some unsatisfiable set of clauses it will not produce the empty clause. Similarly, the replacement rule in plan construction is not complete. The completeness of linear resolution can be assured by a variety of methods, and we have plan construction rules which simulate each of these. Two of these rules are "factoring" and "reduction".

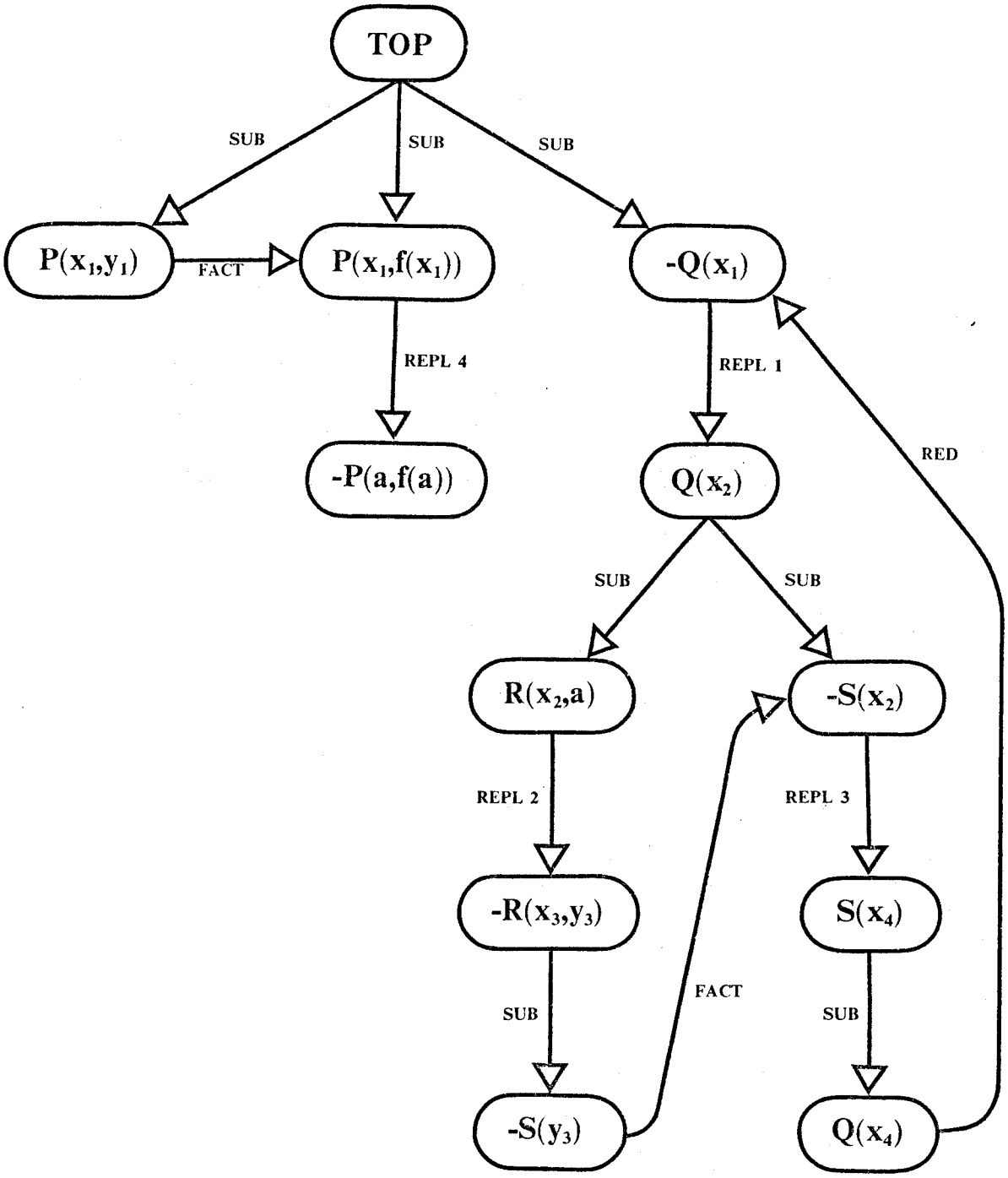
As with replacement, factoring and reduction add new arcs to the plan; however, unlike replacement they introduce only <u>one</u> new arc, and add no new vertices. Factoring is analogous to the familiar factoring rule of resolution, and is applied by selecting an open subproblem of the plan, and directing an arc labelled "FACT" from it to another subproblem of the plan: eventually, of course, these subproblems must be shown to be unifiable. Factoring in plans differs from factoring in resolution in that when a FACT arc is added, the subproblem at its head need not be open. Reduction also

has an analogy in linear resolution, namely, the reduction rule of model-elimination and SL-resolution. This corresponds to the familiar proof technique "reductio ad absurdum" in which a particular hypothesis is shown to imply its negation. To apply reduction, we select an open subproblem u and direct an arc labelled "RED" from it to some subproblem v which is "above" u in the underlying rooted tree: that is, there must be a walk from v to u consisting entirely of REPL and SUB arcs. For the reduction to apply, we must verify that u and the negation of v are unifiable. As for REPL arcs, subproblems at the tail of RED and FACT arcs are said to be closed.

In figure 3.3, we illustrate a closed plan obtained from the plan of figure 3.2 by applying these two rules.

Note that plans containing RED and FACT arcs are not trees.

There is one further rule for constructing plans called "ancestor replacement", which is a variation of the simple replacement rule we have already described. This allows us to close a subproblem by replacement using a clause deduced earlier in the proof, rather than a clause from S. In order to apply this we must have some means of extracting such clauses from the plan. This leads us to the definition of a special kind of subgraph of a plan called a "subplan". The important feature of subplans is that, although they cannot necessarily be constructed from S using the rules we have described, they have the same underlying rooted tree structure as plans. That is, each subplan contains the vertex TOP, the REPL and SUB arcs of the subplan form a rooted tree, and if a REPL arc is in the subplan, then



A closed plan for the set of clauses of example 3.1.1.

Figure 3.3

so are all the SUB arcs associated with that REPL arc. To apply ancestor replacement, some subplan is extracted from the plan, and the set of open subproblems of this subplan is used as though it were a clause in S for closing a subproblem of the plan by replacement.

We present now the formal definition of deduction plans, followed by the proofs of soundness and completeness of the various deduction systems based on them.

3.2: Formal Definition of Plans

3.2.1: Definition: Let S be a set of clauses. A pair (G,λ) is a deduction plan (or simply a plan) for S where G is a graph and λ is a mapping of E(G), called the labelling, into the set {REPL,SUB,RED,FACT} iff it is defined inductively as follows:

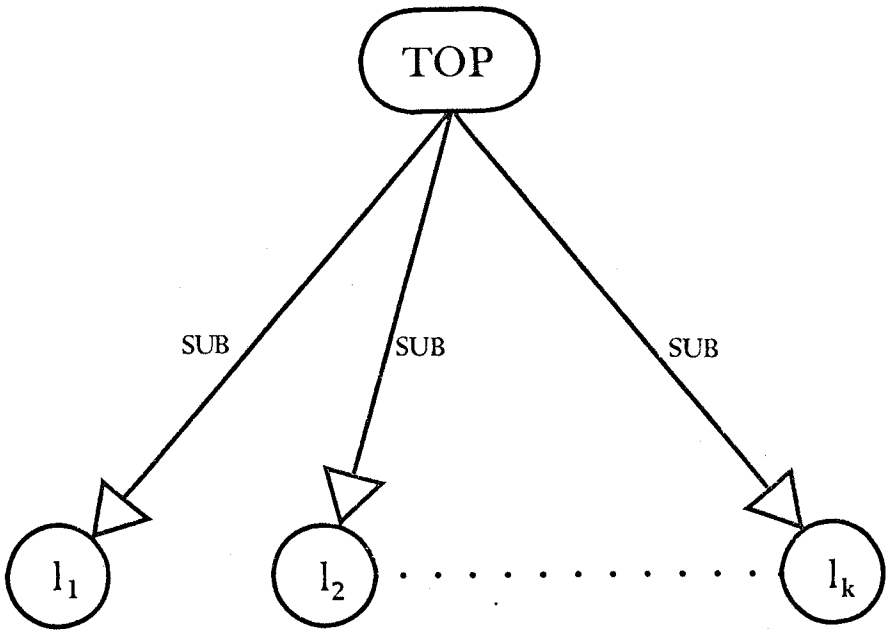
3.2.1.1: Basis

$$G = IND(\{(TOP,x) | x \in C\})$$

where $C \in S$ and TOP is an arbitrary but fixed object such that $TOP \notin L \times I$.

For $e \in E(G)$: $\lambda(e) = SUB$

This initial plan is called basic and C is called the top clause. (See Fig. 3.4).



A pictorial representation of a basic plan with top clause $\{l_1,\ldots,l_k\}$.

Figure 3.4

3.2.1.2: Induction:

Let us assume that a plan (G,λ) for S is defined. Before we proceed with the presentation of the induction rules we introduce a few definitions.

We shall denote counter images of REPL,SUB,RED,FACT under λ respectively as REPL(G), SUB(G), RED(G), FACT(G). They form a partition of E(G) and by specifying them we can define the mapping λ.

Let $u,v \in V(G)$. u is a direct ancestor of v iff there is a walk from u to v containing only arcs of $REPL(G) \cup SUB(G)$. u is a strong ancestor of v iff u is a direct ancestor and every walk from TOP to v passes through u. If u is a direct (strong) ancestor of v, then v is said to be a direct (strong) descendant of u.

Let H be a subgraph of G. $(H,\lambda|E(H))$ is a subplan of (G,λ) for S iff for each $x \in V(H)$ the following hold:

(I) all direct ancestors of x in G belong to V(H)

(II) if $(y,x) \in REPL(G)$ then every arc of G which has x as its head or tail belongs to E(H).

In future, if there is no danger of ambiguity, we will refer to a plan or subplan (G,λ) simply as G. Also, we will say that H is a subplan for S if there exists a plan G for S, and H is a subplan of G for S. When the context ensures that there will be no ambiguity, we will say H is a subplan of G, or simply H is a subplan.

Let H be a subplan of G. The set $\{v \mid \exists x \in V(H)$ such that $(x,v) \in SUB(H)\}$ is called the set of subproblems of H and denoted s(H). s(H) is partitioned into the subset of open subproblems with outdegree = 0 and its complement, closed subproblems. These sets are denoted respectively os(H) and cs(H).

Now we may proceed with the induction. A pair (G',λ') is a plan called an inductive extension of (G,λ) iff it is defined by one of the following 3 rules:

Rule 1: Replacement

$$G' = IND(E(G) \cup \{(u,v)\} \cup \{(v,x) \mid x \in C' - \{v\}\})$$

where C' is a variant of a copy of a nonempty clause C such that C' and V(G) are separated, $u \in os(G)$ and $v \in C'$.

For $e \in E(G')$:

$$\lambda'(e) = \begin{cases} \textbf{REPL} & \textbf{if } e = (u,v) \\ \text{SUB} & \text{if } e \in \{(v,x) \mid x \in C' - \{v\}\} \\ \boldsymbol{\lambda(e)} & \textbf{otherwise} \end{cases}$$

This rule is divided into two subcases:

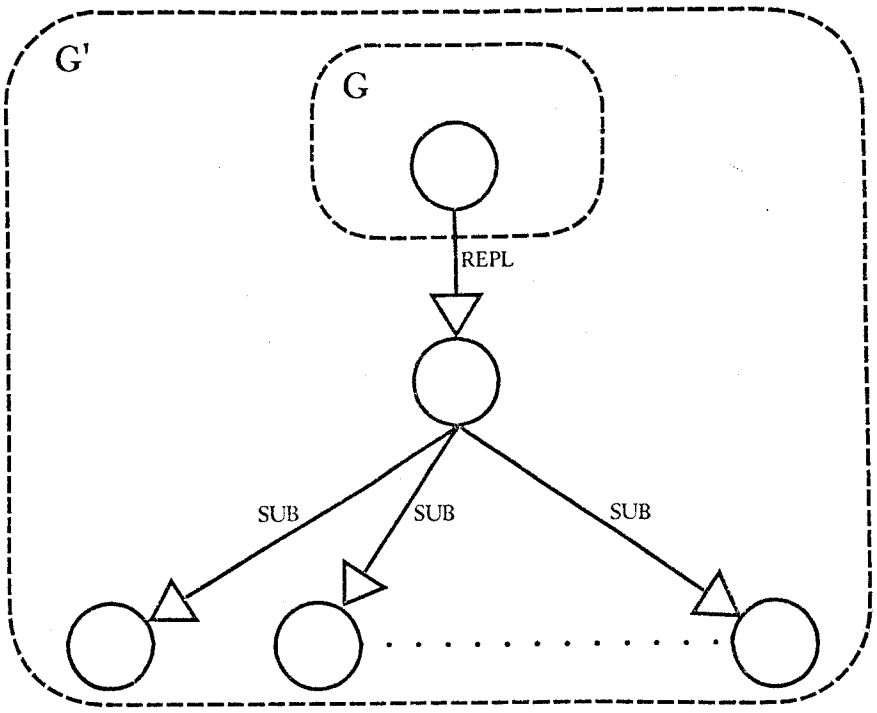
(A) Simple Replacement:

$$C \in S$$

(B) Ancestor Replacement:

$$C = os(H) \text{ where } H \text{ is a subplan of } G.$$

When Rule 1 is applied we say that u is replaced through v by $C' - \{v\}$. (See Figure 3.5).



A representation of a plan G'
obtained from G by replacement.

Figure 3.5

Rule 2: Reduction

$$G' = IND(E(G) \cup \{(v,u)\})$$

where $v \in os(G)$ and u is a strong ancestor of v. For $e \in E(G')$:

$$\lambda'(e) = \begin{cases} RED & \text{if } e = (v,u) \\ \lambda(e) & \text{otherwise} \end{cases}$$

In this case we say that v is reduced to u.

Rule 3: Factoring

$$G' = IND(E(G) \cup \{(u,v)\})$$

where $u \in os(G)$, $v \in s(G) - \{u\}$ and

(I) if there is a walk from v to u it must contain at least one arc from RED(G)

(II) if $(x,y) \in RED(G)$ then y remains a strong ancestor of x in G'.

For $e \in E(G')$:

$$\lambda'(e) = \begin{cases} FACT & \text{if } e = (u,v) \\ \lambda(e) & \text{otherwise} \end{cases}$$

This rule can be divided into 2 subcases:

(A) Simple Factoring

$$v \in os(G)$$

In this subcase both conditions (I) and (II) are automatically satisfied.

(B) Back Factoring

$$v \in cs(G).$$

When Rule 3 is applied we say that u is factored to v.

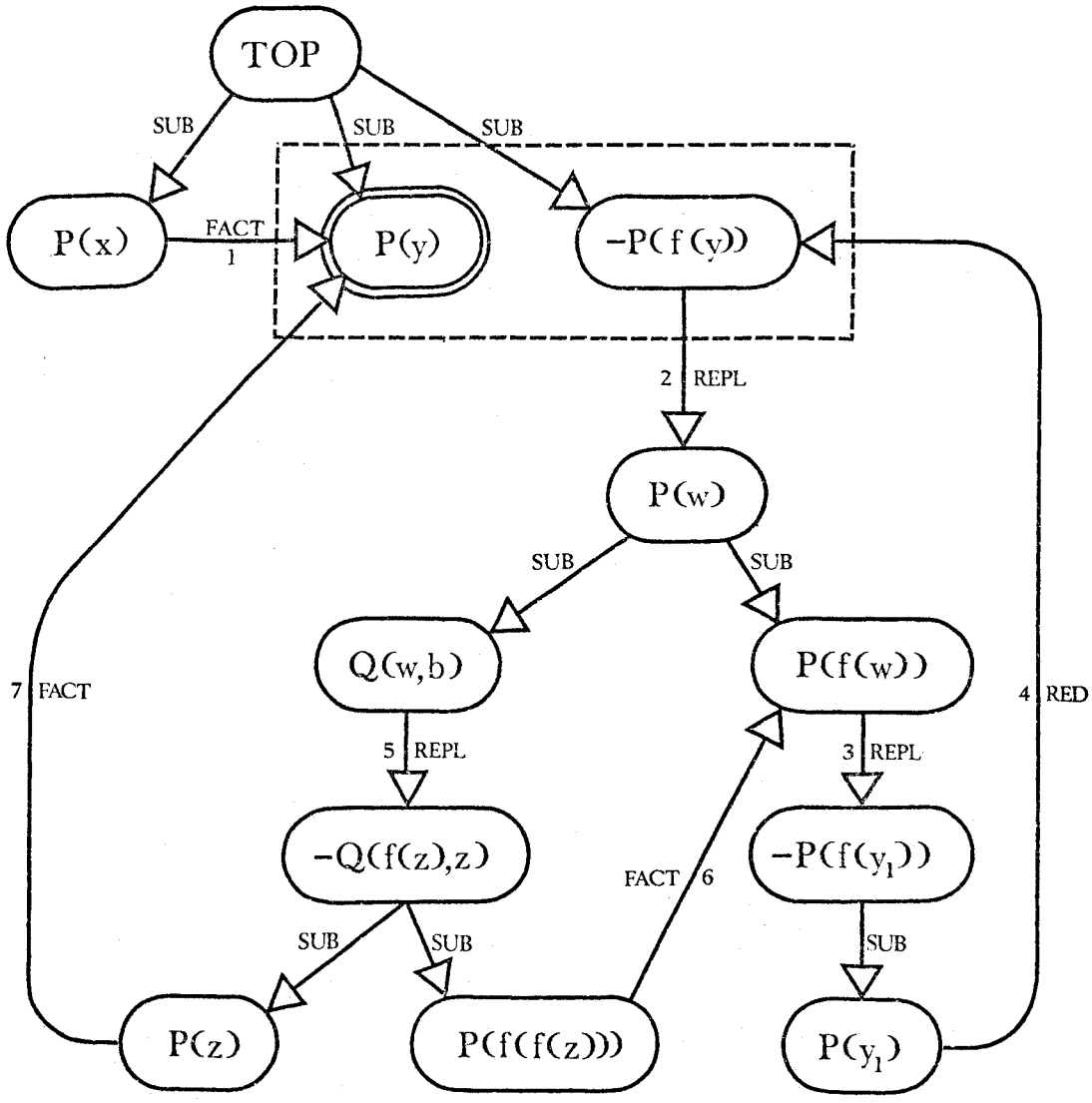
Figure 3.6 illustrates the violation of conditions (I) and (II) on factoring.

We now present an example to illustrate the construction of a plan.

3.2.2: Example: Let S be the set of clauses:

$$\{\ \{P(x),\ P(y),\ -P(f(y))\},$$
$$\{P(w),\ Q(w,b),\ P(f(w))\},$$
$$\{-Q(f(z),z),\ P(z),\ P(f(f(z)))\}\ \}$$

where b is a constant. Figure 3.7 illustrates a plan G for S. Each arc from REPL(G) $\cup$ RED(G) $\cup$ FACT(G) is labelled with an integer specifying a derivation of G. We will use such integer labels in future examples. Note that the REPL arc numbered 3 is an ancestor replacement using the boxed clause. G has only one open subproblem, which is the vertex with a double outline.



A plan G for the set S of clauses of example 3.2.2.

Figure 3.7

Note that in the above example the arc REPL 2 could be constructed before the arc FACT 1. It is clear, therefore, that a particular plan may be constructed in more than one way. We will find it necessary at times to refer to a specific order of construction of a plan: we use derivations for this purpose.

3.2.3: Definition: If G is a plan and $D = (G_0,\ldots,G_n)$ is a sequence of plans such that G_0 is basic, $G_n = G$ and for $1 \leq i \leq n$, G_i is an inductive extension of G_{i-1}, then D is called a derivation of G of length n.

3.3: Structural Properties of Plans

In this section we note some obvious consequences of the above definitions, prove some technical results, and prove some lemmas concerning the relationships between the plan construction rules.

3.3.1: Consequences of Definitions

We will denote the set $REPL(G) \cup FACT(G) \cup RED(G)$ by $SOL(G)$.

(a) In every subplan, there are vertices which are not subproblems: TOP is one of these, and the others are those vertices through which subproblems are replaced.

(b) Closed subproblems are former open subproblems each of which has been closed by one application of a rule.

(C) Suppose a plan G has a derivation of length n.

Since one application of a rule closes exactly one subproblem, and adds exactly one arc to SOL(G), we note that $||cs(G)|| = ||SOL(G)|| = n$, where $||\ ||$ denotes set cardinality, and that every derivation of G has the same length. We note that every derivation of G begins with the same basic plan. Consequently, for any plan G we can specify a derivation either as an ordering of $cs(G)$, or equivalently, as an ordering of $SOL(G)$. This leads to some notational devices which we will use frequently, despite their initially ambiguous appearance: namely, to define some new plan G_2 by applying a rule to G_1, we may write:

$$cs(G_2) = cs(G_1) \cup \{x\}$$
$$\text{or } SOL(G_2) = SOL(G_1) \cup \{(x,y)\}$$
$$\text{or } FACT(G_2) = FACT(G_1) \cup \{(x,y)\}$$
$$\ldots \text{ etc.}$$

when it is obvious from the context exactly how x is to be closed, and exactly what vertices are to be added.

(d) If G is a plan, there is no closed walk in G with all its arcs in $REPL(G) \cup SUB(G)$.

(e) The only rule that adds new vertices to a plan is replacement.

3.3.2: Lemma: If G is a plan, $w \in cs(G)$ and:

$G' = IND(E(G) - \{(u,v) | (u,v) \in E(G)$
and either $u = w$
or (u or v is a direct descendant of w)\})

then G' is a subplan of G.

Proof: Note that G' is a subgraph of G.

Suppose now that G' is not a subplan of G: we have three cases:

(i) Suppose for some $x \in V(G')$ that:

$$(x,y) \in SUB(G) - SUB(G')$$

Since $(x,y) \notin SUB(G')$

either $x = w$, which is impossible since w is a subproblem and x is not.

or x is a direct descendant of w, in which case $x \notin V(G')$; a contradiction.

or y is a direct descendant of w, so that x is also a direct descendant of w, and again $x \notin V(G')$; a contradiction.

(ii) Suppose for some $x \in V(G')$ that:

$$(y,x) \in REPL(G) - REPL(G')$$

Since $(y,x) \notin REPL(G')$

either $y = w$, so that x is a direct descendant of w

or y is a direct descendant of w, so that x is also a direct descendant of w

or x is a direct descendant of w

In each case $x \notin V(G')$; a contradiction.

(iii) Finally, suppose for some $x \in V(G')$ that y is a direct ancestor of x and $y \notin V(G')$. In this case, y must be a direct descendant of w, so that x is a direct descendant of w, and therefore $x \notin V(G')$; a contradiction.

Therefore G' is a subplan of G. □

3.3.3: Lemma: If G is a plan, $(u,v) \in RED(G)$, and $(w,z) \in FACT(G)$, then:

(i) v is a strong ancestor of u in G

(ii) if there is a walk from z to w in G, it must contain an arc in RED(G).

Proof: Let $D = (G_0,\ldots,G_n)$ be a derivation of G.

(i) Obviously v is a direct ancestor of u in G.

Suppose there is a walk in G from TOP to u: we must show that this walk passes through v. The proof is by induction on the number of arcs in RED(G) that lie on this walk.

Basis: Suppose the walk contains no arcs in RED(G). Let w be the last subproblem on this walk to be closed in the derivation D, then for some k, $0 < k \leq n$:

$$cs(G_k) = cs(G_{k-1}) \cup \{w\}$$

We have two cases to consider:

(a) w = u. In this case, the walk from TOP to u exists in G_{k-1}, since all the subproblems on the walk except u are closed in G_{k-1}. Hence v is a strong ancestor of u in G_{k-1}, so that the walk must pass through v.

(b) $w \neq u$. Suppose w is closed by replacement through z by $z_1,\ldots,z_m$ then for some i, $1 \leq i \leq m$, z_i lies on the walk and is an open subproblem of G_k. This is a contradiction since w is the last subproblem on the walk to be closed. w must therefore be closed by factoring. However, the walk from TOP to u exists in G_k and $(u,v) \in RED(G_k)$, so by condition (II) on factoring, v is a strong ancestor of u in G_k. Therefore the walk must pass through v.

Induction: Assume the result holds for walks containing $< m$ arcs of RED(G), and that the walk we are considering contains m arcs of RED(G). Let this walk be $TOP, w_1, w_2, \ldots, w_r$, where $u = w_r$. Suppose that w_k is the first subproblem on this walk closed by reduction. Then $TOP, w_1, w_2, \ldots, w_k$ is a walk in G from TOP to w_k with no arcs in RED(G). By the induction hypothesis, this walk must pass through w_{k+1}, so for some s, $1 \leq s \leq k$, $w_s = w_{k+1}$. Hence $TOP, w_1, \ldots, w_s, w_{k+2}, \ldots, w_r$ is a walk from TOP to u in G containing m-1 arcs of RED(G). By the induction hypothesis, this walk must pass through v; therefore the original walk must pass through v. □

3.3.4: Corollary: Suppose H is a subplan of G, H is a plan, $(x,y) \in FACT(G)$, $(u,v) \in RED(G)$, and $x,u \in os(H)$.

(i) If H' is the subplan of G defined by:

$$SOL(H') = SOL(H) \cup \{(x,y)\}$$

then H' is a plan provided that $y \in s(H)$

(ii) If H'' is the subplan of G defined by:

$$SOL(H'') = SOL(H) \cup \{(u,v)\}$$

then H'' is a plan.

Proof: Lemma 3.2.4 ensures that the conditions for closing either x or u are satisfied in H. □

We will now characterize the relationships between plans, derivations, and subplans. First we note that a subplan is not necessarily a plan, as illustrated by the following example.

3.3.5: Example: Consider the plan G_2 for the set of clauses $\{\{v_1,v_2\}, \{v_3,v_4\}\}$, where G_2 has the derivation (G_0,G_1,G_2) defined as follows:

$$V(G_0) = \{TOP, v_1, v_2\}$$
$$SUB(G_0) = \{(TOP,v_1), (TOP,v_2)\}$$

$$V(G_1) = V(G_0) \cup \{v_3,v_4\}$$
$$SUB(G_1) = SUB(G_0) \cup \{(v_3,v_4)\}$$
$$REPL(G_1) = REPL(G_0) \cup \{(v_1,v_3)\}$$

$$V(G_2) = V(G_1) \cup \{v_2',v_4'\}$$
$$SUB(G_2) = SUB(G_1) \cup \{(v_4',v_2')\}$$
$$REPL(G_2) = REPL(G_1) \cup \{(v_2,v_4')\}$$

where $\{v_2',v_4'\}$ is a copy of a variant of $\{v_2,v_4\}$.

Now H is a subplan of G_2, where:

$$V(H) = \{TOP, v_1, v_2, v_4', v_2'\}$$
$$E(H) = \{(TOP,v_1), (TOP,v_2), (v_4',v_2'), (v_2,v_4')\}$$

However, H is not a plan. This is illustrated in figure 3.8.

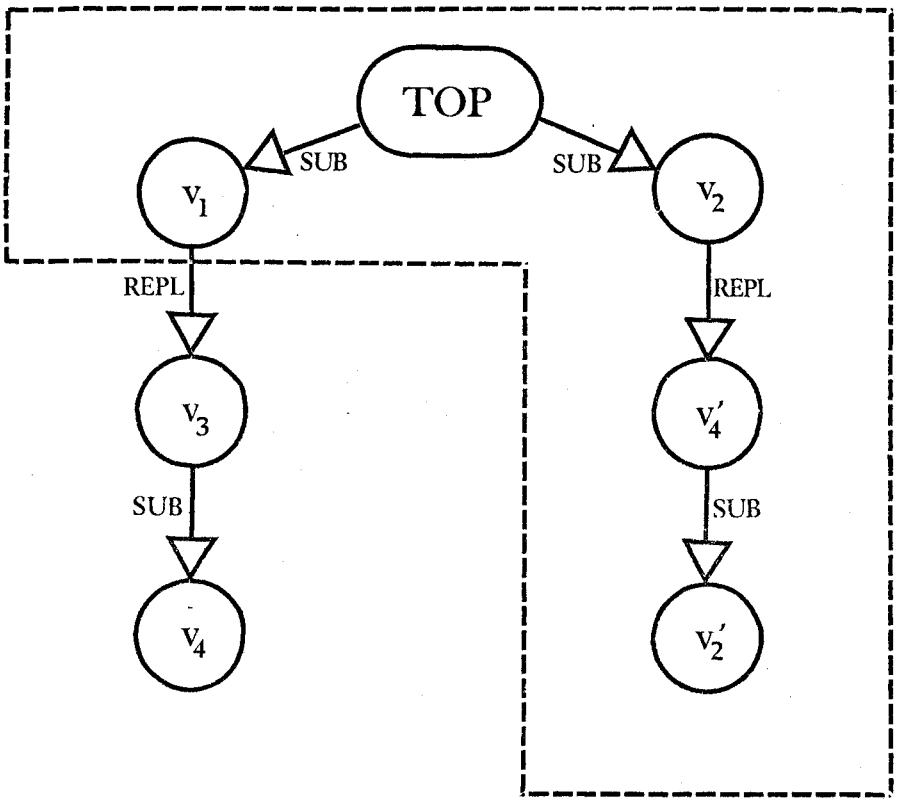


Figure 3.8

As the reader may have realised, situations such as that described in the above example can arise only in the presence of the ancestor replacement rule. This is proved in the following lemma.

3.3.6: Lemma: If G is a plan constructed using rules (1)A, (2) and (3) only, then every subplan of G is a plan which can be constructed using (1)A, (2), and (3) only.

Proof: Suppose G_m is a subplan of G, where $m = ||SOL(G_m)||$. We will show that there exists a subplan G_{m-1} such that $||SOL(G_{m-1})|| = m-1$, and G_m is a plan if G_{m-1} is a plan. We consider two cases.

Case (a): If $RED(G_m) \cup FACT(G_m)$ is not empty, define G_{m-1} by:

$$V(G_{m-1}) = V(G_m)$$
$$E(G_{m-1}) = E(G_m) - \{(x,y)\}$$

where $(x,y) \in RED(G_m) \cup FACT(G_m)$. Now $||SOL(G_{m-1})|| = m-1$, and by corollary 3.3.4, if G_{m-1} is a plan then G_m is a plan.

Case (b): Suppose $RED(G_m) \cup FACT(G_m)$ is empty. Let $x_1,\ldots,x_k$ be the closed subproblems of G_m in order of their closure in some derivation D of G. Suppose x_k is replaced through y by $C - \{y\}$, where C is a variant of some clause in S. Now since G_m is a subplan, $C \subseteq V(G_m)$. Also $C - \{y\} \subseteq os(G_m)$, since otherwise $x_i \in C - \{y\}$ for some $i < k$ since x_i is closed before x_k in the derivation D. Hence we can define G_{m-1} by:

$$V(G_{m-1}) = V(G_m) - C$$
$$E(G_{m-1}) = E(G_m) - \{(x_k,y)\} - \{(y,z) | z \in C - \{y\}\}$$

Now $||SOL(G_{m-1})|| = m-1$ and if G_{m-1} is a plan, G_m is clearly a plan.

Since $SOL(G_m)$ is finite, a finite number of applications of this process must eventually yield a subplan having no arcs of SOL(G). The only such subplan is G_0, the basic plan of G. Hence G_m is a plan. □

There exists a special relationship between derivations and subplans which are themselves plans, as shown by the following lemma and its corollary.

3.3.7: Lemma: If G and G' are plans, and G' is a subplan of G, then every derivation $(G_0,\ldots,G_m)$ of G' may be extended to a derivation $(G_0,\ldots,G_n)$ of G, where $n \geq m$.

Proof: Suppose $G_m \neq G$. We show that there exists a plan G_{m+1} which is a subplan of G, and is an inductive extension of G_m.

Since $G_m \neq G$, the set $os(G_m) \cap cs(G)$ is not empty. Let $x_1,\ldots,x_k$ be the members of this set in the order of their closure in some derivation D of G. We define G_{m+1} by:

$$cs(G_{m+1}) = cs(G_m) \cup \{x_1\}$$

It remains to prove that G_{m+1} is a plan. We have four cases:

(a) Suppose x_1 is closed by simple replacement in G. Then the conditions for closing x_1 are trivially satisfied in G_m, so that G_{m+1} is a plan.

(b) Suppose x_1 is closed by reduction, then corollary 3.3.4 guarantees that G_{m+1} is a plan.

(c) Suppose x_1 is closed by factoring to y in G.

Suppose $y \notin V(G_m)$, then there must exist some $x_i \in os(G_m)$ such that x_i is a direct ancestor of y, and x_i is closed by replacement in G. But x_i must be closed before x_1 in every derivation of G, in particular in D, contrary to the ordering imposed on $os(G_m) \cap cs(G)$. Therefore $y \in V(G_m)$, so corollary 3.3.4 guarantees that G_{m+1} is a plan.

(d) Suppose x_1 is closed by ancestor replacement using some subplan H of G. We must show that H is a subplan of G_m. Suppose the contrary, then we have two cases:

(i) $\exists y \in V(H) - V(G_m)$

In this case, $\exists x_i \in os(G_m)$ such that x_i is a direct ancestor of y and is closed by replacement in G. But x_i is closed in H, and therefore is closed before x_1 in any derivation of G. This contradicts the ordering imposed on $os(G_m) \cap cs(G)$.

(ii) $\exists e \in E(H) - E(G_m)$, say $e = (y,z)$

If $e \in SUB(G)$, then $z \in V(H) - V(G_m)$, which we have already shown to be impossible.

If $e \in SOL(G)$, then since $V(H) \subseteq V(G_m)$ by case (i), y is an open subproblem of G_m, and so $y = x_i$ for some $i > 1$. But y must be closed before x_1 in any derivation of G; this contradicts the ordering imposed on $os(G_m) \cap cs(G)$.

H is therefore a subplan of G_m, so the conditions for closing x_1 by ancestor replacement are satisfied in G_m. Therefore G_{m+1} is a plan.

Since $cs(G)$ is finite and $||cs(G_{m+1})|| > ||cs(G_m)||$, a finite number of such extensions must eventually result in a derivation for G. □

3.3.8: Corollary: A subgraph H of a plan G is a plan if and only if there is a derivation $(G_0,\ldots,G_n)$ of G such that $H = G_m$ for some $m \leq n$.

We will now investigate the relationships between the rules from the point of view of the structure of plans.

3.3.9: Reduction and Factoring

(i) In the absence of factoring we can replace the phrase "strong ancestor" by the phrase "direct ancestor" in the definition of reduction.

(ii) In the absence of reduction, condition (II) on factoring may be omitted.

In the absence of ancestor replacement, factoring is equivalent to simple factoring. In order to establish this result we must first prove the following technical lemma.

3.3.10: Lemma: If G is a plan such that $FACT(G) \neq 0$ and for all $(x,y) \in FACT(G)$, y is neither open, nor closed by reduction; then $\exists (x,y) \in FACT(G)$ such that y is closed by replacement and no direct descendant of y is closed by factoring.

Proof: Suppose the contrary; that is:

(A) for all $(x,y) \in FACT(G)$,

either y is closed by factoring,

or y is closed by replacement and a direct descendant of y is closed by factoring.

We first prove that for any integer $n \geq 1$, there is a walk of length n in G such that the last arc in the walk is in $FACT(G)$ and no arc of the walk is in $RED(G)$. The proof is by induction on n.

Basis: $n = 1$. Fact is not empty, so there exists $(x_1,x_2) \in FACT(G)$. The walk from x_1 to x_2 consisting of this single arc has the required properties.

Induction: Suppose there exists a walk $(x_1,x_2),\ldots,(x_{n-1},x_n)$ with the required properties. Now $(x_{n-1},x_n) \in FACT(G)$ by the induction hypothesis; so by the hypothesis (A), we have two cases:

either x_n is closed by factoring to z, say. Then $(x_1,x_2),\ldots,(x_{n-1},x_n)$, (x_n,z) is a walk of length $> n$ with the required properties.

or x_n is closed by replacement, and some direct descendant w of x_n is closed by factoring to some z. Therefore, there is a walk from x_n to w containing only arcs of $REPL(G) \cup SUB(G)$, and a walk from w to z consisting of the single arc $(w,z) \in FACT(G)$. Appending these two walks to the end of the walk $(x_1,x_2),\ldots,(x_{n-1},x_n)$, we obtain a walk of length $> n$ with the required properties.

Since this holds for any integer, and $V(G)$ is finite, there exists a walk in G of length $||V(G)|| + 1$ with no arcs in $RED(G)$. Such a walk must encounter some vertex more than once: hence there is a closed walk in G containing no arcs of $RED(G)$. By 3.3.1(d), some arc (x,y) on this walk is in $FACT(G)$. Consequently, there exists $(x,y) \in FACT(G)$, and a walk from y to x with no arcs in $RED(G)$. This contradicts lemma 3.2, thereby disproving hypothesis (A), and establishing the result. □

We may now establish the relationship between ancestor replacement and factoring.

3.3.11: Lemma: If G is a plan constructed using rules (1)A and (3), then G can be constructed using (1)A and (3)A.

Proof: We will show that there exists a derivation $(G_0,\ldots,G_n)$ of G such that for some $m \leq n$, G_m is obtained from G_{m-1} by simple factoring, and if $m < n$, then for $i > m$, G_i is derived from G_{i-1} by simple replacement. Since G_{m-1} is a plan constructed using (1)A and (3) only, and $||FACT(G_{m-1})|| = ||FACT(G_m)|| - 1$, a finite number of applications of this process will yield a derivation of G in which all factorings are simple.

If $FACT(G)$ is empty, there is nothing to prove, so we suppose the contrary. We have two cases to consider.

Case (a): Suppose for some $(x,y) \in FACT(G)$, y is open.

Obviously this factoring is simple. Define G_{n-1} by:

$$V(G_{n-1}) = V(G)$$
$$E(G_{n-1}) = E(G) - \{(x,y)\}$$

Then G_{n-1} is a subplan of G by lemma 3.3.2, and hence a plan by lemma 3.3.6. If $(G_0,\ldots,G_{n-1})$ is a derivation of G_{n-1}, then $(G_0,\ldots,G_n)$, where $G_n = G$, is a derivation of G with the required properties.

Case (b): Suppose that for every $(x,y) \in FACT(G)$, y is closed.

(A) Suppose that for every $(x,y) \in FACT(G)$,

if y is closed by replacement,

then either some direct descendant of y is closed by factoring,

or some subproblem z is factored to a direct descendant of y.

We show that hypothesis (A) leads to a contradiction. To do this, we show that, for any integer k, there exist two sequences of subproblems, $x_1, x_2, \ldots, x_k$, and $y_1, y_2, \ldots, y_k$ such that for all i, where $1 \leq i < k$:

(i) $(x_i, y_i) \in FACT(G)$

(ii) y_i is closed by replacement.

(iii) y_{i+1} is a direct descendant of y_i

(iv) no direct descendant of y_i is closed by factoring.

These sequences are constructed inductively as follows:

Basis: Since for every $(x,y) \in FACT(G)$, y is closed, lemma 3.2.8 ensures that $\exists (x,y) \in FACT(G)$ such y is closed by replacement, and no direct descendant of y is closed by factoring. Let $x_1 = x$ and $y_1 = y$. Then x_1 and y_1 obviously satisfy the conditions (i) to (iv).

Induction: Suppose a suitable sequence of length $k-1$ has been constructed. Now y_{k-1} is closed by replacement, and no direct descendant of y_{k-1} is closed by factoring, by the induction hypothesis. Therefore by the hypothesis (A), some subproblem z is factored to a direct descendant y of y_{k-1}. Let $x_k = z$ and $y_k = y$; then:

(i) $(x_k, y_k) \in FACT(G)$

(ii) y_k is closed by replacement since it is a direct descendant of y_{k-1}, and by the induction hypothesis, no direct descendant of y_{k-1} is closed by factoring.

(iii) y_k is a direct descendant of y_{k-1}.

(iv) no direct descendant of y_k is closed by factoring, since this would imply that y_{k-1} has a direct descendant closed by factoring.

Hence sequences of length k exist with the required properties.

Now since such sequences exist to any length, and $s(G)$ is finite, there exists a pair of sequences of length $||s(G)|| + 1$: in this case, $y_i = y_j$ for some $i < j$, and y_j is a direct descendant of y_i. This implies the existence of a closed walk in G, all the arcs of which are in $REPL(G) \cup SUB(G)$, contrary to 3.3.1(d). Thus hypothesis (A) is disproved, and there exists $(x,y) \in FACT(G)$ such that y is closed by replacement, no direct descendant of y is closed by factoring, and there is no subproblem z factored to a direct descendant of y.

We now define:

$V(G_m) = V(G) - \{z | z$ is a direct descendant of $y\}$

$E(G_m) = E(G) - \{\{(w,z) | z$ is a direct descendant of $y\}$

It is easy to see that this definition is equivalent to the definition of the subgraph G' in lemma 3.3.2, so by that lemma, G_m is a subplan of G, and therefore is a plan by lemma 3.3.6. Also $(x,y) \in FACT(G)$ and y is open, so by case(a) there is a derivation $(G_0,\ldots,G_m)$ such that G_m is obtained from G_{m-1} by simple factoring. Now by lemma 3.3.7, this derivation can be extended to a derivation for G, and obviously G_i is obtained from G_{i-1} by simple replacement, for $i > m$. Finally, from case(a) we have $||FACT(G_{m-1})|| = ||FACT(G_m)|| - 1 = ||FACT(G)|| - 1$. This completes the proof. □

3.3.12: Corollary: If G is a plan constructed using rules (1)A, (2) and (3) only, then G can be constructed using rules (1)A, (2) and (3)A only.

Proof: We define a subgraph H of G by:

$$V(H) = V(G)$$

$$E(H) = E(G) - RED(G)$$

One application of lemma 3.3.2 and one application of lemma 3.3.6 for each arc of RED(G) removed, proves that H is a plan. But H is constructed using rules (1)A and (3) only, so by lemma 3.3.11, there exists a derivation $(G_0,\ldots,G_m)$ of H, constructed using rules (1)A and (3)A only. By lemma 3.3.7, this derivation can be extended to a derivation $(G_0,\ldots,G_n)$ for G, where for $i \geq m$, G_i is obtained from G_{i-1} by reduction. $(G_0,\ldots,G_n)$ is a derivation of G constructed using rules (1)A, (2) and (3)A only. □

3.4: Constrained Plans

So far plans have been discussed purely as syntactic objects. We now ascribe some meaning to them by relating them to predicate calculus using the notion of "constrained plan" in which we keep track of the literals which must be unified in order to validate the rules used in constructing the plan.

3.4.1: Definition: A constrained plan is an ordered pair (G,C) where G is a plan and C is a function from $E(G)$ into the set of sets of unordered pairs of literals, defined inductively as follows:

Basis: If G is a basic plan

$$C(e) = \emptyset \quad \text{for} \quad e \in E(G)$$

Induction: If (G,C) is a constrained plan then (G',C') is a constrained plan where G' is an inductive extension of G and C' is defined as follows:

(a) if $e \in E(G)$ then:

$$C'(e) = C(e)$$

(b) if $e \in SUB(G') - E(G)$ then:

$$C'(e) = \emptyset$$

(c) if $e = (u,v) \in SOL(G') - E(G)$ then:

(i) if G' is obtained from G by simple replacement or reduction:

$$C'(e) = \{\{|u|, \neg|v|\}\}$$

(ii) if G' is obtained from G by factoring:

$$C'(e) = \{\{|u|,|v|\}\}$$

(iii) if G' is obtained from G by ancestor replacement of u through v by $os(K)'\gamma = \{v\}$ where $os(K)'$ is a copy of $os(K)$, K is a subplan of G and γ is a renaming such that $V(K)\gamma$ has no variable in common with $V(G)$ then:

$$C'(e) = \{\{|u|,\neg|v|\}\} \cup \bigcup_{e \in E(K)} C(e)\gamma$$

We shall say that (G',C') is an inductive extension of (G,C) by the appropriate rule. If (G,C) is a constrained plan we define constrained derivation in the obvious way as a sequence of constrained plans.

3.4.2: Definition: If (G,C) is a constrained plan, then an ordered pair (H,D) is a constrained subplan of (G,C) iff H is a subplan of G and $D = C|E(H)$. If (H,D) is a constrained subplan we define:

$$D(H) = \bigcup_{e \in E(H)} D(e)$$

and call this the set of constraints of (H,D).

The following example shows that the constraint set of a plan is not unique.

3.4.3: Example: Consider the set of clauses

$$S = \{\{Q(x),P(x)\},$$
$$\{-P(a)\},$$
$$\{-Q(y),R(y)\},$$
$$\{-R(z),-Q(b)\},$$
$$\{Q(u)\}\}$$

where a and b are constants.

Figure 3.9 illustrates a plan which corresponds with two different constrained plans. The replacement 4 could be a simple replacement, in which case we have a constrained plan with constraint set $\{\{x,a\},\{x,y\},\{y,z\},\{b,u\}\}$. This replacement could also be an ancestor replacement using the outlined subplan and the renaming $\gamma = \{(x,u)\}$: in this case the constraint set is $\{\{x,a\},\{x,y\},\{y,z\},\{b,x\}\gamma,\{x,a\}\gamma\}$. Note that in the first case, the constraints are unifiable, but in the second case they are not.

The situation illustrated in the above example is due purely to the use of ancestor replacement.

3.4.4: Lemma: If (G,C) and (G,D) are constrained plans constructed without ancestor replacement, then C = D.

The proof is left is the reader, and follows immediately from definition 3.4.1.

We now introduce the key notion of correctness.

3.4.5: Definition: A constrained subplan (H,C) is said to be correct if C(H) is unifiable, in which case we denote the most general unifier of C(H) by $\theta((H,C))$, and call the clause $os(H)\theta((H,C))$ the clause deduced by (H,C). Note that every constrained subplan of a correct constrained plan is correct.

3.4.6: Convention: In the rest of this paper, we consider only constrained plans. To simplify the discussion when there is no danger of ambiguity we will therefore use the words "plan", "subplan" and "derivation" to mean "constrained plan" etc. Also we will abbreviate (H,C) simply as H.

3.4.7: Example: Consider the plan G of example 3.2.2 (Figure 3.7). Figure 3.10 lists the constraints constituting C(G): each constraint in this list is labelled with the integer corresponding to the arc in SOL(G) from which it originates (see Figure 3.7). The constraint set C(G) is unifiable, and its most general unifier is:

$$\theta(G) = \{(x,a), (y,a), (z,a), (w,f(a)), (y_1,f(a))\}$$

(iii) G'_{j+1} is derived from G'_j by ancestor replacement and simple factoring only.

(iv) there is a substitution α_j such that:

$$z\theta(G_j) = z\theta(G'_j)\circ\alpha_j \quad \text{for all} \quad z \in s(G_j)$$

This sequence is constructed as follows:

(A) $G'_m = G_m$

(B) Suppose the construction is complete up to G'_j. G_{j+1} is generated from G_j by reducing some $x \in os(G_j)$ to some direct ancestor y. Let H be defined by:

$$V(H) = V(G'_j) - \{z \mid z \text{ is a direct descendant of } y\}$$

$$E(H) = E(G'_j) - \{(w,z) \mid w \text{ or } z \text{ is a direct descendant of } y\}$$

By lemma 3.3.2, H is a subplan of G'_j. Also $os(H) = \{y', x'_1, \ldots, x'_k\}$, where $\{x'_1, \ldots, x'_k\} \subseteq os(G'_j)$. We now generate a sequence of graphs $(G_j^0, \ldots, G_j^k)$ as follows:

(a) Let γ be a renaming such that $V(H)\gamma$ is a variant of $V(H)$. G_j^0 is the plan obtained from G'_j by replacing $x \in os(G'_j)$ through $y\gamma$ by $\{x_1\gamma, \ldots, x_k\gamma\}$, where $\{y\gamma, x_1\gamma, \ldots, x_k\gamma\}$ is a copy of $\{y'\gamma, x'_1\gamma, \ldots, x'_k\gamma\}$ which is separated from $V(G'_j)$.

(b) For $i \in \{1, \ldots, k\}$ G_j is the plan defined by:

$$FACT(G_j^i) = FACT(G_j^{i-1}) \cup \{(x_i\gamma, x'_i)\}$$

Let $G'_{j+1} = G_j$; it remains to show that G'_{j+1} satisfies the required conditions.

(i) G'_{j+1} is clearly a plan, so we need only show that $G(G'_{j+1})$ is unifiable. First we show that $\{|x|\theta(G'_j), \neg|y|\theta(G'_j)$ is unifiable. Since

3.5: Further Results on Equivalence of Rules

In this section we present two lemmas establishing the semantic equivalence of certain subsets of the plan construction rules.

3.5.1: Lemma: If G is a correct plan for S generated by rules (1)A and (2) only, then there exists a correct plan G' for S generated by rules (1) and (3)A only, such that:

$$os(G)\theta(G) = os(G')\theta(G')\circ\alpha$$

for some substitution α.

Proof: If $RED(G) = \emptyset$ there is nothing to prove, so we assume the contrary. Let K be defined by:

$$V(K) = V(G)$$

$$E(K) = E(G) - RED(G)$$

One application of lemma 3.3.2 and one application of lemma 3.3.6 for each arc of $RED(G)$ removed, proves that K is a plan which can be constructed using (1)A only; so by lemma 3.3.7, there is a derivation $(G_0,\ldots,G_n)$ such that $G_m = K$ for some $m < n$; for $0 < i \leq m$ G_i is derived from G_{i-1} by simple replacement; and for $m < i \leq n$, G_i is derived from G_{i-1} by reduction. By lemma 3.4.4, $G_n = G$.

We now construct a sequence $(G'_m,G'_{m+1},\ldots,G'_n)$ such that for $j \geq m$:

(i) G'_j is a correct plan,

(ii) $os(G'_j) = os(G_j)$

$C(G_{j+1}) = C(G_j) \cup \{\{|x|, \neg|y'|\}\}$ is unifiable, by lemma 2.3.3 and the fact that $|y'| = |y|$, $\{|x|\theta(G_j), \neg|y|\theta(G_j)\}$ is unifiable, so by the induction hypothesis condition (iv), $\{|x|\theta(G'_j)\circ\alpha_j, \neg|y|\theta(G'_j)\circ\alpha_j\}$ is unifiable and therefore $\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\}$ has a unifier $\alpha_j \circ \text{mgu}\{|x|\theta(G_j), \neg|y|\theta(G_j)\}$. Therefore there is a substitution β such that:

(1) $\ldots\alpha_j \circ \text{mgu}\{|x|\theta(G_j), \neg|y|\theta(G_j)\} = \text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\}\circ\beta$

We now proceed to show that $C(G'_{j+1})$ is unifiable.

(2) $\ldots C(G'_{j+1}) = C(G'_j) \cup C(H)\gamma \cup \{\{|x|, \neg|y\gamma|\}\}$

$$\cup \{\{|x_i\gamma|, |x'_i|\} \mid i \in \{1,\ldots,k\}\}$$

Let $\delta = \gamma^{-1} \circ \theta(G'_j) \circ \text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\}$

We show that δ unifies each set in the above expression (2) for $C(G'_{j+1})$.

$$C(G'_j)\delta = (C(G'_j)\gamma^{-1})(\theta(G'_j)\circ \text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\})$$

$$= (C(G'_j)\theta(G'_j))\text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\}$$

since none of the replaced variables of γ^{-1} occur in $C(G'_j)$.

So since $\theta(G'_j)$ unifies $C(G'_j)$, δ unifies $C(G'_j)$.

$$(C(H)\gamma)\delta = (C(H)(\gamma\circ\gamma^{-1}))(\theta(G'_j)\circ \text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\})$$

$$= (C(H)(\gamma^{-1})(\theta(G'_j)\circ\text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\})$$

by 2.4.3

$$= (C(H)\theta(G'_j)\circ\text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\}$$

since none of the replaced variables of γ^{-1} occur in $C(H)$

So since $C(H) \subseteq C(G'_j)$, $\theta(G'_j)$ unifies $C(H)$, so that δ unifies $C(H)$.

$$\{\{|x|, \neg|y\gamma|\}\}\delta = \{\{|x|\gamma^{-1}, \neg|y|\gamma^{-1}\}\}(\theta(G'_j) \circ \text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\})$$

by 2.4.3

$$= \{\{|x|, \neg|y|\}\}(\theta(G'_j) \circ \text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\})$$

since none of the replaced variables of γ^{-1} occur in x or y

$$= \{\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\}\}\text{mgu}\{|x|\theta(G'_j), \neg|y|\theta(G'_j)\}$$

Therefore δ unifies $\{\{|x|, \neg|y\gamma|\}\}$.

Finally, for each $i \in \{1,\ldots,k\}$:

$$\{|x_i\gamma|, |x'_i|\}\alpha = \{|x_i|\gamma^{-1}, |x'_i|\gamma^{-1}\}(\theta(G'_j) \circ \text{mgu}\{|x|\theta(G'_j), |y|\theta(G'_j)\})$$

by 2.4.3

Therefore δ unifies $\{|x_i\gamma|, |x_i|\}$, since $|x_i| = |x'_i|$.

Hence $C(G'_{j+1})$ is unifiable so that G'_{j+1} is correct.

(ii) $os(G'_{j+1}) = os(G'_j) - \{x\}$

$= os(G_j) - \{x\}$

$= os(G_{j+1})$

(iii) Since $\{x'_1,\ldots,x'_k\} \subseteq os(G'_j)$, all the factorings performed in deriving G'_{j+1} are simple.

(iv) **In the proof of (i) above, we have shown that**

$\gamma^{-1} \circ \theta(G'_j) \text{mgu} \{ |x|\theta(G'_j), \neg|y|\theta(G'_j)\}$ **unifies** $C(G'_{j+1})$, **so for some substitution** τ:

(3) $\ldots\theta(G'_{j+1}) \circ \tau = \gamma^{-1} \circ \theta(G'_j) \circ \text{mgu} \{ |x|\theta(G'_j), \neg|y|\theta(G'_j)\}$

Let β be the substitution defined in equation (1) in part (i) above, and let $\alpha_{j+1} = \tau \circ \beta$, then if $z \in s(G_{j+1})$:

$$z\theta(G'_{j+1}) \circ \alpha_{j+1} = z(\theta(G'_{j+1}) \circ \tau) \circ \beta$$

$$= z\gamma^{-1} \circ \theta(G'_j) \circ \text{mgu} \{ |x|\theta(G'_j), \neg|y|\theta(G'_j)\} \circ \beta$$

from (3) above

$$= z\theta(G'_j) \circ \text{mgu} \{ |x|\theta(G'_j), \neg|y|\theta(G'_j)\} \circ \beta$$

since none of the replaced variables of γ^{-1} occur in z

$$= z\theta(G'_j) \circ \alpha_j \circ \text{mgu} \{ |x|\theta(G_j), \neg|y|\theta(G_j)\}$$

by (1) in (i) above

$$= z\theta(G_j) \circ \text{mgu} \{ |x|\theta(G_j), \neg|y|\theta(G_j)\}$$

by induction hypothesis, condition (iv), and since $z \in s(G_{j+1}) = s(G_j)$

But $C(G_{j+1}) = C(G_j) \cup \{\{x, \neg y\}\}$

So by lemma 2.3.3:

$$\theta(G_{j+1}) = \theta(G_j) \circ \text{mgu} \{ |x|\theta(G_j), \neg|y|\theta(G_j)\}$$

$\therefore$ $z\theta(G_{j+1}) = z\theta(G'_{j+1}) \circ \alpha_{j+1}$

The sequence having been constructed, let $G' = G'_n$ and $\alpha = \alpha_n$, then:

$$os(G) = os(G')$$

$$\text{and } z\theta(G) = z\theta(G')\circ\alpha \quad \text{for all} \quad z \in s(G)$$

$$\therefore os(G)\theta(G) = os(G')\theta(G')\circ\alpha \quad \square$$

3.5.2: Lemma: If G is a closed, correct plan for S generated by rules (1)A and (2) only, then there exists a closed, correct plan G' for S generated by rules (1) and (3)B only.

Proof: If $RED(G) = \emptyset$ there is nothing to prove, so we assume the contrary. Let K and the derivation $(G_0,\ldots,G_n)$ be defined as in the proof of lemma 3.5.1, and let H be defined by:

$$V(H) = V(G_m) - \{ z \mid \exists (x,y) \in RED(G) \text{ such that } z \text{ is a direct descendant of } y\}$$

$$E(H) = E(G_m) = \{ (w,z) \mid \exists (x,y) \in RED(G) \text{ such that either } w \text{ or } z \text{ is a direct descendant of } y\}$$

Again, lemmas 3.3.2 and 3.3.6 ensure that H is a subplan of G.

Now suppose $x \in os(G_m)$, then $(x,y) \in RED(G)$ for some direct ancestor y of x; but x is then a direct descendant of y, so that $x \notin V(H)$. Hence $os(H) \subseteq cs(G_m)$.

We now construct a sequence of graphs $(G'_m, G'_{m+1}, \ldots, G'_n)$ such that for $j \geq m$:

(i) G'_j is a correct plan.

(ii) $os(G'_j) = os(G_j)$

(iii) G'_{j+1} is derived from G'_j by ancestor replacement and back factoring only.

(iv) there is a substitution α_j such that:

$$z\theta(G_j) = z\theta(G'_j)\circ\alpha_j \quad \text{for all} \quad z \in s(G_j)$$

(v) G_m is a subplan of G'_j.

This sequence is constructed as follows:

(A) $G'_m := G_m$

(B) Suppose the construction is complete up to G'_j. G_{j+1} is generated from G_j by reducing some $x \in os(G_j)$ to some direct ancestor y. Now $y \in os(H)$, since y is at the head of an arc of $RED(G)$, and all such vertices lost their direct descendants in the construction of H. Suppose $os(H) = \{y', x'_1, \ldots, x'_k\}$. We now generate a sequence of graphs $(G^0_j, \ldots, G^k_j)$ as follows:

(a) Let γ be a renaming such that $V(H)\gamma$ is a variant of $V(H)$. G^0_j is the plan obtained from G'_j by replacing $x \in os(G'_j)$ through $y\gamma$ by $\{x_1\gamma, \ldots, x_k\gamma\}$ where $\{y\gamma, x_1\gamma, \ldots, x_k\gamma\}$ is **a copy of $\{y'\gamma, x'_1\gamma, \ldots, x'_k\gamma\}$ which is separated from $B(G'_j)$.**

(b) For $i \in \{1, \ldots, k\}$, G^i_j is the plan defined by:

$$FACT(G^i_j) = FACT(G^{i-1}_j) \cup \{(x_i\gamma, x'_i)\}$$

Let $G'_{j+1} = G_j^k$; it remains to show that G'_{j+1} satisfies the required conditions.

(i), (ii) and (iv) are proved analogously to the corresponding parts of lemma 3.5.1.

(iii) Now since $os(H) \subseteq cs(G_m)$, then by condition (iv), $os(H) \subseteq cs(G'_j)$. Hence all the factorings performed in deriving G'_{j+1} from G'_j are back factorings.

(v) G_m is a subplan of G'_j and hence of G'_{j+1}.

The sequence having been constructed, let $G' = G'_n$, then $os(G') = os(G) = \emptyset$, and G' is correct. □

These two lemmas show that reduction may be replaced by ancestor replacement and factoring. In the first lemma we see that with simple factoring we can simulate reduction in that for every plan obtained using reduction, we can construct a plan with the same open subproblems. The second lemma shows that such a close correspondence does not necessarily exist using backfactoring.

4. Soundness and Completeness

In this section we present the basic result, linking the notion of satisfiability with that of closedness and correctness. Soundness is established directly by considering the relationship between a model for a set of clauses and a plan for the set. The proof of completeness relies on the completeness of the resolution principle.

4.1 Soundness

4.1.1 Definition: If G is a subplan, and γ is any substitution, we define $TOP\gamma = TOP$ and:

$$E(G)\gamma = \{(x\gamma, y\gamma) \mid (x,y) \in E(G)\}$$

Also, we denote the graph $\langle V(G)\gamma, E(G)\gamma \rangle$ by $G\gamma$, and call $G\gamma$ an instance of G. $G\gamma$ is a ground instance of G iff for every $x \in V(G)$, $x\gamma$ is a ground instance of x. If γ is a renaming $G\gamma$ is a variant of G iff $V(G)\gamma$ is a variant of $V(G)$. If G is a plan for a set S of clauses, then clearly so is every variant of G if we define the labelling of the variant in the obvious way.

4.1.2: Lemma: Let G be a correct plan for a set S of clauses, H a subplan of G, Σ a model for S, and $H\theta(H)\circ\gamma$ a ground instance of $H\theta(H)$: then there exists a walk $(x_0,x_1),\ldots,(x_{n-1},x_n)$ in H containing no arcs in $RED(H)$, such that $x_0 = TOP$, $x_n \in os(H)$, and $\Sigma(|x_i|\theta(H)\circ\gamma) = T$ for every $x_i \in s(H)$, $0 < i \leq n$.

Proof: Let D be a derivation of G. We prove the result by induction on the number of ancestor replacements in D.

Basis: Suppose D is constructed without ancestor replacement. We construct the walk recursively as follows:

(i) The top clause $\{z|(TOP,x) \in SUB(H)\}$ of H is some clause in S, and therefore contains some element x_1 such that $\Sigma(|x_1|\theta(H)\circ\gamma) = T$. (TOP,x_1) is the first arc in the walk.

(ii) Suppose the walk has been constructed up to the arc (x_{i-1},x_i), and that x_i is a subproblem. Suppose $(x_i,y) \in RED(H)$.

either the walk from TOP to x_i contains no arcs of FACT(H), in which case all direct ancestors of x_i must lie on the walk, so y in particular must lie on the walk.

or the walk from TOP to x_i contains arcs in FACT(H). In this case, let (x_j,x_{j+1}) be the first arc on the walk in FACT(H), where $j < i$, then:

either that part of the walk from x_{j+1} to x_i passes through y,

or by lemma 3.3.3, y is a direct ancestor of x_j, so that part of the walk from TOP to x_j passes through y.

In any event, y must lie on the walk.

$$\therefore \Sigma(|y|\theta(H)\circ\gamma) = T$$

But $\Sigma(|y|\theta(H)\circ\gamma) = \Sigma(\neg|x_i|\theta(H)\circ\gamma)$

$$= F$$

which is a contradiction. Therefore x_i is not closed by reduction in H.

Hence we have three cases to consider:

(a) $x_i \in os(H)$, in which case $n = i$, and we have the required walk.

(b) x_i is closed by factoring in H, say $(x_i,z) \in FACT(H)$, in which case:

$$z\theta(H) = x_i\theta(H)$$

$$\therefore \Sigma(|z|\theta(H)\circ\gamma) = \Sigma(|x_i|\theta(H)\circ\gamma)$$

$$= T$$

Let $x_{i+1} = z$; (x_i,x_{i+1}) is then the next arc in the walk.

(c) x_i is closed by simple replacement through y in H; then we define $x_{i+1} = y$, so that $(x_i,x_{i+1}) \in REPL(H)$. In this case $\{x_{i+1}\} \cup \{x|(x_{i+1},x) \in SUB(H)\}$ is a copy of a variant of a clause in S, and therefore:

$$\Sigma([\{|x_{i+1}|\} \cup \{|x| \; |(x_{i+1},x) \in SUB(H)\}]\theta(H)\circ\gamma) = T$$

But $|x_{i+1}|\theta(H) = \neg|x_i|\theta(H)$

$$\therefore \Sigma(|x_{i+1}|\theta(H)\circ\gamma) = \Sigma(\neg|x_i|\theta(H)\circ\gamma)$$

$$= F$$

Hence $\exists z \in \{x|(x_{i+1},x) \in SUB(H)\}$ such that:

$$\Sigma(|z|\theta(H)\circ\gamma) = T.$$

We define $x_{i+2} = z$. Then (x_{i+1},x_{i+2}) are the next two arcs in the walk, and satisfy the required conditions.

Suppose this construction does not terminate as in case (a), then since V(H) is finite, the process must generate a walk which passes through some vertex of H twice. In the latter case, there exists a closed walk in H containing no arcs in RED(H). By 3.3.1(d), however, no closed walk can

consist entirely of arcs in REPL(H) ∪ SUB(H). Therefore some arc (x_j, x_{j+1}) on this closed walk must belong to FACT(H): then there is a walk from x_{j+1} to x_j containing no arcs in RED(H), contrary to lemma 3.3.3. Hence the construction must terminate as in case (a).

Induction: Assume the result for derivations with fewer ancestor replacements than D. We then construct the required walk in H exactly as in the basis of the proof, except that we have one more case to consider when extending the walk, as follows:

(d) x_i is closed by ancestor replacement, using a variant $K\alpha$ of some subplan K of G. Suppose x_i is replaced through $y\alpha$. Let $x_{i+1} = y\alpha$, then $\{y\} \cup \{w | (x_{i+1}, w\alpha) \in SUB(H)\}$ is a copy of os(K).

Let $D = (G_0, \ldots, G_m)$, then for some $k \leq m$:

$$cs(G_k) = cs(G_{k-1}) \cup \{x_i\}$$

G_{k-1} is a correct plan for S, constructed using fewer ancestor replacements than are used in the construction of G, and K is a subplan of G_{k-1}. So by the induction hypothesis:

(1) If $K\theta(K)\circ\tau$ is a ground instance of $K\theta(K)$, then $\exists z \in os(K)$ such that:

$$\Sigma(z|\theta(K)\circ\tau) = T$$

Also we have:

$$C(K)\alpha \subseteq C(H)$$

$$\therefore \theta(H) \text{ unifies } C(K)\alpha$$

$$\therefore \alpha\circ\theta(H) \text{ unifies } C(K)$$

Therefore:

(2).....$\alpha\circ\theta(H) = \theta(K)\circ\beta$

for some substitution β

$H\theta(H)\circ\gamma$ is a ground instance of $H\theta(H)$. Suppose variables occur in $(K\alpha)\theta(H)\circ\gamma$; this is possible since K is not necessarily a subplan of H. In this case, let γ_1 be some substitution such that no variables occur in $(K\alpha)\theta(H)\circ\gamma\circ\gamma_1$; then:

(3).....$x\theta(H)\circ\gamma\circ\gamma_1 = x\theta(H)\circ\gamma$ if $x \in V(H)$

Now since $(K\alpha)\theta(H)\circ\gamma\circ\gamma_1$ contains no variables, by applying (2), we see that $K\theta(K)\circ(\beta\circ\gamma\circ\gamma_1)$ is a ground instance of $K\theta(K)$, so by (1):

$$\Sigma(|z|\theta(K)\circ(\beta\circ\gamma\circ\gamma_1)) = T \quad \text{for some } z \in os(K)$$

Now
$$\Sigma(|z|\theta(K)\circ(\beta\circ\gamma\circ\gamma_1)) = \Sigma(|x_{i+1}|\theta(H)\circ(\gamma\circ\gamma_1))$$

by applying (2)

$$= \Sigma(\neg|x_i|\theta(H)\circ(\gamma\circ\gamma_1))$$

$$= \Sigma(\neg|x_i|\theta(H)\circ\gamma)$$

by (3)

$$= F$$

$$\therefore |z| \neq |y|$$

$$\therefore |z| \in \{|w||(x_{i+1},w\alpha) \in SUB(H)\}$$

Let $u \in \{w|(x_{i+1},w\alpha) \in SUB(H)\}$ be such that $|u| = |z|$, and let $x_{i+2} = u\alpha$.

Then $(x_{i+1},x_{i+2}) \in SUB(H)$

and $\in \Sigma(|x_{i+2}|\theta(H)\circ\gamma) = \Sigma((|z|\alpha\ \theta(H)\circ\gamma\circ\gamma_1)$

$$= \Sigma(|z|\theta(K)\circ(\beta\circ\gamma\circ\gamma_1))$$

$$= TT$$

$(x_i,x_{i+1}),(x_{i+1},x_{i+2})$ are then the next two arcs on the walk and satisfy the required conditions. □

This result allows us to establish soundness as follows.

4.1.3: THeorem: The Soundness of Plans

If there exists a closed, correct plan for a set S of clauses, then S is unsatisfiable.

Proof: Let G be a closed, correct plan for S, and suppose S has a model. Then by lemma 4.1.2, there is a walk from TOP to y in G such that $y \in os(G)$, contradicting the fact that G is closed. Therefore S has no model. □

4.2: Completeness

To establish the completeness of plans, we consider first only those plans with derivations constructed using simple replacement and reduction. We prove the completeness of these restricted plans using a standard technique: the result is obtained first for the ground case, and is then generalised. For the ground case our proof relies on the completeness of resolution.

4.2.1: Definition: If A and B are separated ground clauses and u is a literal such that

$$u = |p| = \neg|q| \text{ for some } p \in A \text{ and } q \in B,$$

then the set:

$$C = (A - \{x \in A \mid |x| = u\}) \cup (B - \{x \in B \mid |x| = \neg u\})$$

is called the resolvent of A and B on u. Since A and B are separated, C is a clause.

4.2.2: Definition: If S is a set of ground clauses, then we define a set of ground clauses $R(S)$ by:

$$R(S) = \{c \mid \exists A,B,u \text{ such that } C \text{ is a resolvent of } A' \text{ and } B' \text{ where } A' \text{ and } B' \text{ are separated copies of } A \text{ and } B \text{ respectively}\}$$

We also define inductively:

$$S^{(0)} = S$$
$$S^{(i+1)} = S^{(i)} \cup R(S^{(i)}) \quad \text{for } i \geq 0$$

Although our definition of "clause" differs from the usual in that we allow multiple occurrences of literals, it is obvious that the following theorem, due to Robinson [14], holds for our definition of clauses and resolution.

4.2.3: Theorem (Robinson)

A set of ground clauses S is unsatisfiable iff for some $k \geq 0$, $\square \in S^{(k)}$.

4.2.4: Convention: We now temporarily restrict our attention to derivations constructed using rules (1)A and (2) only, so to simplify the discussion, we will use the word "derivation" with this restricted meaning. Similarly we will use the word "plan" to mean a plan with such a restricted derivation. By lemma 3.4.4, the constraint function and constraint set will be independent of the particular derivation we choose for a plan throughout the scope of this convention.

The next lemma (4.2.5) is the key result in proving the completeness of plans for ground clauses. In it, we show that if there is a closed correct plan for the set $S \cup R(S)$, then there is a closed correct plan for S. By applying this result inductively and using theorem 4.2.3, we can then obtain the desired result. The proof of lemma 4.2.5 is inductive. In the inductive step we select some application of the replacement rule which uses a clause C in $R(S)$, and remodel this portion of the plan by performing two replacements using the clauses A and B from S of which C is a resolvent. Unfortunately, replacement and resolution do not correspond

exactly, since replacement operates on only one element from each clause whereas resolution operates on several. Consequently, our remodelling introduces several new open subproblems. These can be closed, however, using copies of other parts of the plan as shown in lemma 4.2.6. Another problem arises in our remodelling: namely, when we have restructured one part of the plan, how do we know that all the old subproblems which existed in the original plan can still be closed in the same way? Again lemma 4.2.6 ensures that they can. Strictly, lemma 4.2.6 should precede lemma 4.2.5 in the presentation; however, in isolation it has no intuitive basis, so we present it after 4.2.5 where it is far more easily understood.

4.2.5: Lemma: Let S be a set of ground clauses. If there exists a closed correct plan for $S \cup R(S)$ then there exists a closed correct plan for S.

Proof: Suppose G is a plan for $S \cup R(S)$ with derivation $(G_0,\ldots,G_m)$ in which there are μ replacements performed using clauses from $R(S)$. Now we will construct a new derivation of a closed correct plan H for $S \cup R(S)$ in which there are $\mu-1$ such replacements. Clearly μ applications of this process will yield a derivation of a closed correct plan for S.

Let k be an integer such that $0 \le k \le m$, G_k is derived from G_{k-1} by replacement of u through v by $C - \{v\}$ where C is a copy of a clause from $R(S)$, and for i such that $k < i \le m$, G_i is derived from G_{i-1} by reduction or replacement using a copy of a clause from S. Since C is a copy of a clause from $R(S)$:

$$C = A' \cup B'$$

where $A = A' \cup \{x_1,\ldots,x_\ell\}$ $(\ell \ge 1)$

and $B = B' \cup \{y_1,\ldots,y_r\}$ $(r \ge 1)$

and A and B are copies of clauses from S. Since G is correct, $|x_i| = \neg|y_j|$ for $1 \le i \le \ell$, $1 \le j \le r$. Note that in the derivation $(G_0,\ldots,G_{k-1})$ there are only $\mu-1$ replacements using clauses from $R(S)$.

We define a derivation $(H_0,\ldots,H_{n_\ell})$ as follows:

(i) $H_i = G_i$ for $0 \le i \le k-1$

(ii) H_k is obtained from H_{k-1} by replacing u through v by $A - \{v\}$. We may assume without losing generality that $v \in A$.

(iii) H_{k+1} is obtained from H_k by replacing x_1 through y_1 by $B - \{y_1\}$.

Note that $\{x_1,\ldots,x_\ell\}$ is separated from $\{y_1,\ldots,y_r\}$ and since the index set is infinite, we may assume that $\{x_1,\ldots,x_\ell,y_1,\ldots,y_r\}$ is separated from $V(G_{k-1})$. The correctness of H_k follows from the fact that $\{|u|,\neg|v|\}$ is a constraint of a correct plan G_k. Also, the correctness of H_{k+1} follows from the fact that $\{|x_1|,\neg|y_1|\}$ is unifiable since $|x_1| = \neg|y_1|$.

This construction is illustrated in figure 4.1.

Now we continue with the construction.

(iv) By applying lemma 4.2.6 where G,H,p and q of the lemma correspond with G_k, H_{k+1}, $a \in V(G_k)$ and $a \in V(H_{k+1})$ where $a \in A' \cup B'$, we close a in H_{k+1}, extending the derivation. Similarly, we apply 4.2.6 to the rest of $A' \cup B'$, extending the derivation to $(H_0,\ldots,H_m)$, where:

$$os(H_m) = \{x_2,\ldots,x_\ell\} \cup \{y_2,\ldots,y_r\}$$

The problem of verifying that the conditions for applying the lemma are satisfied is left to the reader who may find figure 4.1 and the definition of the integer k helpful.

By the definition of k the closure of vertices in $A' \cup B'$ relies only on clauses from S, so that the set S' of the lemma is $\subseteq S$. Therefore the number of replacements using clauses from $R(S)$ remains unchanged.

(v) By $r-1$ consecutive applications of reduction, we close $y_2,\ldots,y_r$ by reducing them to x_1, obtaining plans $H_{m+1},\ldots,H_{m+r-1}$. Correctness follows from the fact that $|x_1| = \neg|y_i|$ for $2 \leq i \leq r$.

Figure 4.2 illustrates the constructions performed in (iv) and (v).

(vi) By applying lemma 4.2.6 where G, H, p and q of the lemma correspond with H_{m+r-1}, H_{m+r-1}, x_1 and x_2, we close x_2, extending the derivation to $(H_0,\ldots,H_{n_\ell})$. Similarly, by ℓ-2 further applications of lemma 4.2.7, we close $x_3,\ldots,x_\ell$ extending the derivation to $(H_0,\ldots,H_{n_\ell})$. In all these applications of lemma 4.2.7, S' in the statement of the lemma is the set of all clauses used in replacing x_1 or direct descendants of x_1. Clearly this set contains no clauses from R(S). H_{n_ℓ} is obviously closed, and is correct by lemma 4.2.7.

To complete the proof, we note that only μ-1 replacements using clauses from R(S) are made in the derivation $(H_0,\ldots,H_{k-1})$: in steps (ii) and (iii) of the construction we use only clauses from S: step (iv) does not introduce any such replacements: step(v) consists solely of reductions: and step (vi) also does not introduce any replacements using clauses from R(S). Therefore $(H_0,\ldots,H_{n_\ell})$ is a derivation of a closed correct plan, constructed with μ-1 replacements using clauses from R(S). □

We now prove the technical result required in the above proof.

4.2.6: Lemma: Let G and H be correct plans for a set of ground clauses S. Suppose there exists $p \in cs(G)$ and there exists $q \in os(H)$ such that:

(i) $|p| = |q|$

(ii) all direct descendants of p are closed in G.

(iii) if z is a direct ancestor of p in G then there exists a direct ancestor w of q in H such that $|z| = |w|$.

Let $S' = \{c | p_1$ or some direct descendant of p_1 is closed by replacement using a copy of $C \in S\}$.

Then any derivation $(H_0,\ldots,H_m)$ of H may be extended to $(H_0,\ldots,H_n)$ where:

(a) H_n is correct

(b) $os(H_n) = os(H) - \{q\}$

(c) if H_{i+1} is derived by replacement from H_i $(m \leq i < n)$ using a clause C, then $C \in S'$.

Proof: Let G' be the subgraph of G which results from removing all the direct descendants of p, and every arc which enters or leaves a direct descendant of p. By lemma 3.3.2, G' is a subplan of G, so by lemma 3.3.7 G has a derivation $(G_0,\ldots,G_r,G_{r+1},\ldots,G_{r+s})$ such that $G_r = G'$. Clearly, for each i such that $1 \leq i \leq S$, if G_{r+i} is obtained by closing $x \in os(G_{r+i-1})$, then x is either p or a direct descendant of p.

Let $(H_0,\ldots,H_m)$ be any derivation of H. We now define a sequence $(H_{m+1},\ldots,H_{m+s})$ of plans and a sequence of functions $(\psi_0,\ldots,\psi_s)$ such that for $0 \leq i \leq s$:

(1) if $i \neq 0$ then H_{m+i} is an inductive extension of H_{m+i-1}, and if the extension is by replacement, the clause used is a copy of a clause in S'.

(2) H_{m+i} is correct.

(3) ψ_i is a bijection from $V_i^{(1)}$ to $V_i^{(2)}$ where:

$$V_i^{(1)} = \{x | x \text{ is a direct descendant of } p \text{ in } G_{r+i}\} \cup \{p\}$$

$$V_i^{(2)} = \{x | x \text{ is a direct descendant of } q \text{ in } H_{m+i}\} \cup \{q\}$$

(4) $os(H_{m+i}) = (os(H_m) - \{q\}) \cup \{\psi_i(x) | x \in os(G_{r+i}) \cap V_i^{(1)}\}$

(5) $|\psi_i(x)| = |x|$ for $x \in V_i^{(1)}$

(6) if $x,y \in V_i^{(1)}$

then $(x,y) \in REPL(G_{r+i})$ implies $(\psi_i(x),\psi_i(y)) \in REPL(H_{m+i})$

and $(x,y) \in SUB(G_{r+i})$ implies $(\psi_i(x),\psi_i(y)) \in SUB(H_{m+i})$

We construct this sequence inductively as follows:

Basis: Since $p \in os(G_r)$ and $q \in os(H_m)$ we have:

$$V_0^{(1)} = \{p\}$$

$$V_0^{(2)} = \{q\}$$

Define $\psi_0(p) = q$

The conditions (1) to (6) obviously hold for $i = 0$.

Induction: Assume the construction in complete up to H_{m+k-1} and ψ_{k-1}. Suppose G_{r+k} is obtained from G_{r+k-1} by closing $x \in os(G_{r+k-1})$. Now $x \in V_{k-1}^{(1)}$, so by induction hypothesis (condition (4)) $\psi_{k-1}(x) \in os(H_{m+k-1})$, and by condition (5), $|\psi_{k-1}(x)| = |x|$

We now have two cases to consider:

(A) Suppose x is closed by reduction to some $y \in V(G_{r+k-1})$. Now either y is a direct ancestor of p, or $y \in V^{(1)}_{k-1}$. In the first case, by condition (iii) in the statement of the lemma, there exists w which is a direct ancestor of q and is therefore a direct ancestor of $\psi_{k-1}(x)$. Consequently, we obtain H_{m+k} from H_{m+k-1} by reducing $\psi_{k-1}(x)$ to w. In the second case, $\psi_{k-1}(y)$ is a direct ancestor of $\psi_{k-1}(x)$, by condition (6) of the induction hypothesis, so we generate H_{m+k} from H_{m+k-1} by reducing $\psi_{k-1}(x)$ to $\psi_{k-1}(y)$. In both cases we define $\psi_k = \psi_{k-1}$.

(B) Suppose x is closed by replacement through x_0 by $x_1,\ldots,x_q$ where $\{x_0,x_1,\ldots,x_q\}$ is a copy of a clause in S'. Let $\{y_0,y_1,\ldots,y_q\}$ be another copy of this clause separated from $V(H_{m+k-1})$, where $|y_i| = |x_i|$ for $0 \le i \le q$. We define H_{m+k} to be the plan that results from closing $\psi_{k-1}(x)$ in H_{m+k-1} by replacement through y_0 by $y_1,\ldots,y_n$; we also define:

$$\psi_k(x) = \begin{cases} \psi_{k-1}(x) & \text{for } x \in V^{(1)}_{k-1} \\ y_i & \text{if } x = x_i \quad (0 \le i \le q) \end{cases}$$

In both the above cases, verifying that conditions (1) to (6) hold for H_{m+k} and ψ_k is straightforward, and is left to the reader.

Now let $n = m + s$, then H_n is a correct plan for S with derivation $(H_0,\ldots,H_n)$, clearly satisfying conditions (a) and (c). As for (b) we note that:

$os(H_n) = (os(H_m) - \{q\}) \cup \{\psi_s(x) | x \in os(G_{r+s}) \cap V_s^{(1)}\}$

by condition (4) on the sequences.

$= os(H_m) - \{q\}$

since $os(G_{r+s}) \cap V_s^{(1)} = \emptyset$

$= os(H) - \{q\}$

Hence (b) is satisfied, and the lemma is proved. □

We now prove the completeness of plans.

4.2.7: Theorem: The Completeness of Plans

Let S be an unsatisfiable set of clauses, then there exists a closed, correct plan for S.

Proof: (A) First we prove the result for ground clauses.

Since S is unsatisfiable, by theorem 4.2.3, $\square \in S^{(k)}$ for some $k \geq 0$. Therefore $(\{TOP\}, \emptyset)$ is a closed, correct plan for $S^{(k)}$. If k is 0 the theorem is proven, otherwise by the definition 4.2.2 and lemma 4.2.5, $S^{(k-1)}$ has a closed, correct plan. A further k-1 applications of lemma 4.2.5 proves the result for S.

(B) We may now prove the general case.

Since S is unsatisfiable, by Herbrand's theorem there exists an unsatisfiable set of ground clauses $T = \{C_1\theta_1, \ldots, C_m\theta_m\}$ where $\theta_1, \ldots, \theta_m$ are substitutions whose terms are from the Herbrand universe; and for $1 \leq i \leq m$, $C_i \in S$ and all the replaced variables of θ_i occur in C_i. Since T is unsatisfiable, by part (A) there exists a closed correct plan for T with derivation $(H_0, \ldots, H_n)$. We now inductively define a derivation $(G_0, \ldots, G_n)$ of a plan for S, and a sequence $\alpha_1, \ldots, \alpha_n$ of substitutions such that for $0 \leq i \leq n$:

(a) $C_{H_i}(H_i) = C_{G_i}(G_i)\alpha_i$

where C_{H_i} and C_{G_i} are the constraint functions of H_i and G_i.

(b) $H_i = G_i\alpha_i$

(c) every replaced variable of α_i occurs in $V(G_i)$ or in S.

Basis: H_0 is basic so that for some $D \in T$:

$$H_0 = IND(\{(TOP,x)|x \in D \in T\})$$

Suppose $D = C_r\theta_r$. We then define G_0 to be the basic plan with top clause C_r, that is:

$$G_0 = IND(\{(TOP,x)|x \in C_r\})$$

We also define:

$$\alpha_0 = \theta_r$$

Conditions (a), (b) and (c) are clearly satisfied.

Induction: Suppose $(G_0,\ldots,G_i)$ and $\alpha_0,\ldots,\alpha_i$ have been defined, where $0 \leq i < n$. We now define G_{i+1} and α_{i+1}: there are two cases to consider.

(1) Suppose H_{i+1} is derived from H_i by replacing $q \in os(H_i)$ through y_1 by $\{y_2,\ldots,y_\ell\}$ where $\{y_1,\ldots,y_\ell\}$ is a copy of clause in T. Then there is a clause $\{x_1,\ldots,x_\ell\} = C_k \in S$ such that $|y_j| = |x_j|\theta_k$ for $1 \leq j \leq \ell$. Let γ be a renaming such that $C_k\gamma$ is a variant of C_k having no variables in common with $V(G_i)$ or S, and every replaced variable of γ occurs in C_r. Now let:

$$C_k' = \{(|x_j|\gamma,i_j)|i_j \text{ is the index of } y_j \text{ and } 1 \leq j \leq \ell\}$$

then C'_k is a variant of C_r which is separated from $V(G_i)$ since it has the same variables as $C_k\gamma$ which has no variables in common with $V(G_i)$, and since C'_k has the same indices as $\{y_1,\ldots,y_\ell\}$, $V(G_i)$ has the same indices as $V(H_i)$, and $\{y_1,\ldots,y_\ell\}$ and $V(H_i)$ are separated. Also, by the induction hypothesis $H_i = G_i\alpha_i$, so there is a vertex $q' \in os(G_i)$ such that $q'\alpha_i = q$. We construct G_{i+1} from G_i by replacing q' through $(|x_1|\gamma,i_1)$ by $C'_k - \{(|x_1|\gamma,i_1)\}$: the conditions for performing this replacement are obviously met. We now define:

$$\alpha_{i+1} = \alpha_i \circ \gamma^{-1} \circ \theta_k$$

Suppose that $v \in V(G_{i+1}) - C'_k$, then $v\alpha_{i+1} = v\alpha_i \circ \gamma^{-1} \circ \theta_k = v\alpha_i$, since $v\alpha_i$ contains no variables. Also, we note that for $1 \leq j \leq \ell$:

$$(|x_j|\gamma)\alpha_{i+1} = (|x_j|\gamma)\alpha_i \circ \gamma^{-1} \circ \theta_k$$

However, $|x_j|\gamma$ has no variables in common with $V(G_i)$ or S, and by the induction hypothesis, all the replaced variables of α_i occur in $V(G_i)$ or S; hence $(|x_j|\gamma)\alpha_i = |x_j|\gamma$. Also, by 2.4.3 $\gamma\circ\gamma^{-1} = \gamma^{-1}$ so that $|x_j|\gamma\circ\gamma^{-1} = |x_j|\gamma^{-1} = |x_j|$ since none of the replaced variables of γ^{-1} occur in x_j. Therefore:

$$(|x_j|\gamma)\alpha_i \circ \gamma^{-1} \circ \theta_k = |x_j|\theta_k = |y_j|$$

$$\therefore\ (|x_j|\gamma,i_j)\alpha_{i+1} = (|y_j|,i_j)$$

$$= y_j$$

Summarizing these results, we have:

$$(*)\quad v\alpha_{i+1} = \begin{cases} v\alpha_i & \text{if} \quad v \in V(G_i) \\ \\ y_j & \text{if} \quad vv = (|x_j|\gamma, i_j) \end{cases}$$

We may now verify conditions (a), (b) and (c) for G_{i+1}, α_{i+1}.

(a) $C_{G_{i+1}}(G_{i+1})\alpha_{i+1} = [C_{G_i}(G_i) \cup \{(|q'|, \neg|x_1|\gamma)\}]\alpha_{i+1}$

$= C_{G_i}(G_i)\alpha_i \cup \{(|q'\alpha_i|, \neg|y_1|)\}$

by (*)

$= C_{H_i}(H_i) \cup \{(|q|, \neg|y_1|)\}$

by the induction hypothesis and the definition of q'

$= C_{H_{i+1}}(H_{i+1})$

(b) is trivial to prove using (*), and is left to the reader.

(c) Since $\alpha_{i+1} = \alpha_i \circ \gamma^{-1} \circ \theta_k$, any replaced variable v of α_{i+1} must be a replaced variable of α_i, γ^{-1} or θ_k. If v is a replaced variable of θ_k then it occurs in S. If v is a replaced variable of γ^{-1}, then it occurs in C'_k and therefore in $V(G_{i+1})$. If v occurs in α_i, then it occurs in $V(G_i)$ or S by the induction hypothesis, and therefore in $V(G_{i+1})$ or S.

(2) Suppose H_{i+1} is derived from H_i by reducing $q \in os(H_i)$ to r. Since $H_i = G_i\alpha_i$ there exist q' and r' such that $r' \in os(G_i)$, q' is a direct ancestor of r', and $q'\alpha_i = q$, $r'\alpha_i = r$. We define G_{i+1} by reducing q' to r', and define $\alpha_{i+1} = \alpha_i$. Verification of conditions (a), (b) and (c) is trivial and is left to the reader.

This completes the induction. To complete the proof it is sufficient to note that conditions (a) and (b) on G_n and the fact that H_n is closed and correct ensures that G_n is closed and correct. □

This ends the scope of convention 4.2.4: however, the completeness result of theorem 4.2.7 holds for plans constructed using all the rules.

4.3: Some Final Remarks

We are now in a position to demonstrate that the restrictive conditions on reduction and factoring are necessary to ensure soundness.

4.3.1: Example: Let S be the set of clauses:

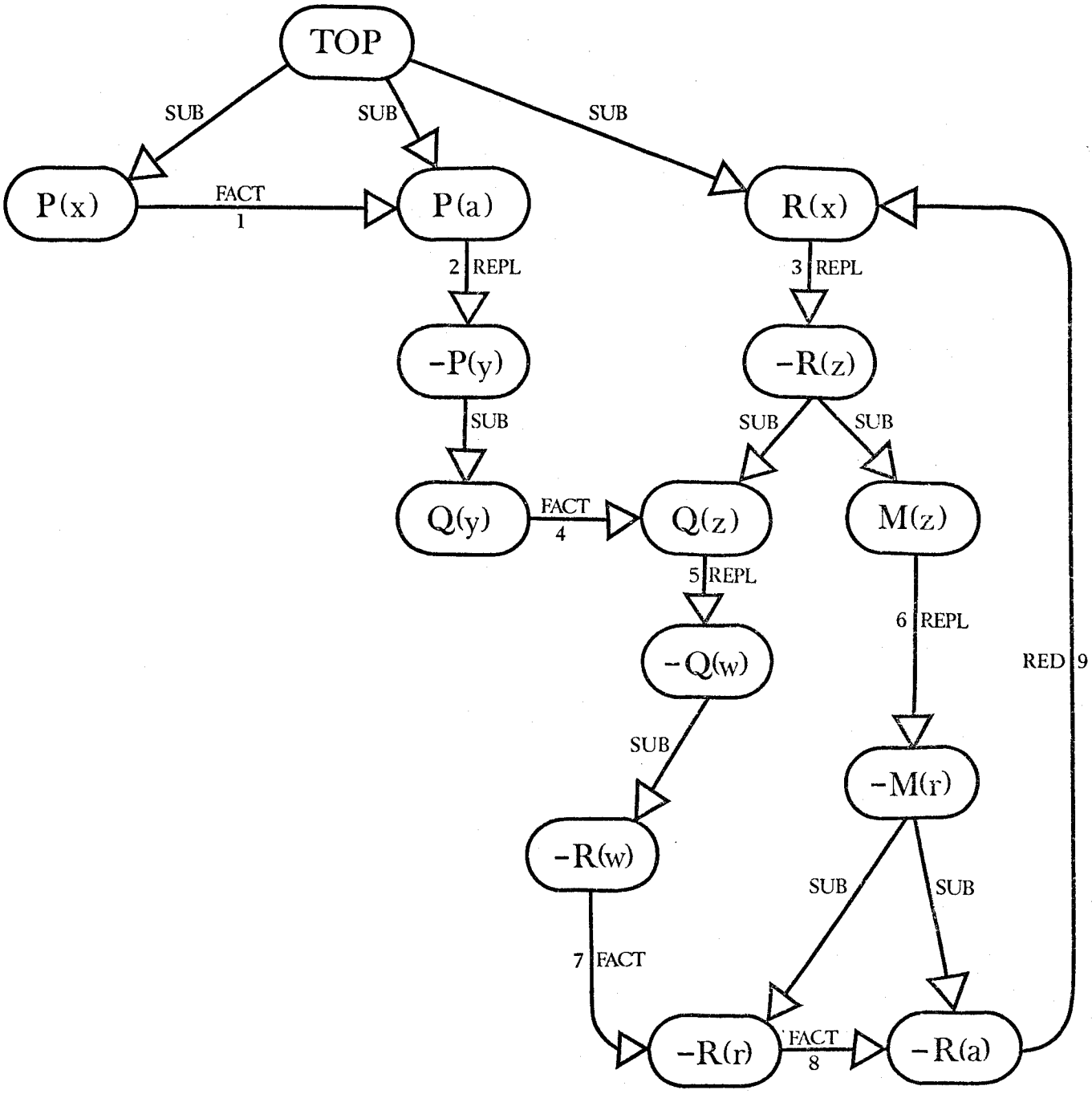
{{P(x), P(a), R(x)},
{-P(y), Q(y)},
{-R(z), Q(z), M(z)},
{-Q(w), -R(w)},
{-M(r), -R(r), -R(a)}}

where a is a constant. S is obviously satisfiable. Consider the graph of figure 4.3: this graph can be constructed using the rules for plan construction with condition (c) on reduction removed, and condition (c) on backfactoring weakened to "y is not an ancestor of x". This graph is closed and is clearly correct, despite the satisfiability of S.

4.3.2: Sound and Complete Subsets of Rules

If R is any subset of the rules for constructing plans, we say that R is sound (complete) if, for every set of clauses S, S is unsatisfiable if (only if) there exists a closed, correct plan for S constructed using the set R of rules.

By theorem 4.1.3 the set of all rules is sound, so obviously any subset is also sound. By theorem 4.2.7 and corollaries 3.5.1 and 3.5.2, the sets {(1)A, (2)}, {(1), (3)A} and {(1), (3)B} are complete, so a superset of



A plan for the set of clauses of example 4.3.1 demonstrating the unsoundness that results when the restrictive conditions on reduction and factoring are removed.

Figure 4.3

any of these sets is also complete. Furthermore, these three sets are minimal in the sense that no subsets of them are complete. This is clear if we observe that sets which do not contain (1)A are not complete; and that neither {(1)A, (3)} nor {(1)} are complete since, for example, neither can generate a closed plan for the unsatisfiable set of clauses {{P(x),P(y)},{-P(x),-P(y)}}. We also note that the subsets {(1)A,(2),(3)} and {(1)A,(2),(3)A} are equivalent in the sense that both generate exactly the same plans for a given set of clauses (corollary 3.3.12). Furthermore, both are equivalent to {(1)A, (2), (3)B} in that they generate the same set of closed plans for a given set of clauses.

5: Practical Considerations

5.1: Constraint processing

We have defined plans and proved the soundness and completeness of various deduction systems based on them. We have not, however, suggested any methods for unifying the set of constraints produced during the construction of a plan.

In a practical theorem-proving system, it would obviously be unwise to attempt to construct a closed, correct plan in the way suggested by the presentation of sections 3 and 4 (that is, by constructing a closed plan, then verifying its correctness) since constraints introduced early in the derivation may be nonunifiable, so that continuing the derivation past the point where these constraints are produced is pointless. Instead, as each open subproblem is closed, the new constraints this closure introduces should be unified with the constraints already produced, to determine whether the new plan is correct. Consequently, to process the constraint set, we require an algorithm which can efficiently unify the constraints on-line as they are produced. This requirement indicates which of the existing unification algorithms we should choose as the basis of our constraint processing system, according to the following argument.

Two formulae may be nonunifiable for two reasons. For example, the formulae $F(G(x))$ and $F(a)$ cannot be unified because of the disagreement between the function symbol G and the constant a. The second type of nonunifiability is exemplified by the two formulae $F(G(x))$ and $F(x)$, which cannot be unified because x occurs in $G(x)$.

A recent unification algorithm of Baxter [1,2] is based on detecting these two types of nonunifiability separately, and accordingly, operates in two stages: first the transformational stage detects nonunifiability due to incompatible function symbols, then the sorting stage checks that no variable is unified with a formula in which it occurs. This requires time proportional to $nG(n)$, where n is the length of the input formulae, and $G(n)$ is an extremely slow-growing function of n. The transformational stage operates in a serial manner on the constraints, and so is particularly suited to our on-line application: the sorting stage is a topological sort of a digraph, and unfortunately, no efficient on-line algorithm is known for this task. However, when a new constraint is added to a previously unified set, only the sorting stage of the algorithm must be completely repeated. By contrast, other unification algorithms combine the transformational and sorting stages, so that complete reprocessing must be done following the addition of a new constraint. A recent algorithm of Paterson and Wegman [13], although of linear time complexity is of this latter type, and hence is not suited to our purposes. In fact, because of its two-stage structure, Baxter's algorithm appears to be the only one which satisfies our requirement for efficient on-line operation.

5.2: Deduction plans and linear deduction

To each linear deduction rule there corresponds a rule for plan construction; however, one of our rules, backfactoring, has no equivalent in existing deduction systems. Backfactoring requires that a record is kept of subproblems that have been solved. The linear systems which have a

reduction rule are the only ones which keep a record of some solved subproblems, but those which are kept are ancestors of the rightmost literal of a chain and so cannot be used in factoring. Hence any factoring in a linear deduction system is simple.

An interesting property of the factoring rule for plan construction is that in any complete subset of the rules which contains factoring, completeness is preserved regardless of which factoring rule we use; so we can actually limit ourselves to factoring only to subproblems which have been closed. This suggests strategies for choosing clauses for use in replacement according to what closed subproblems are available for backfactoring.

Of the three minimal complete subsets of the rules, {(1)A, (2)} corresponds to the ME-deduction system of Loveland [8, 9, 11], and the SL-resolution system of Kowalski and Kuehner [7]. The set {(1), (3)A}, although similar to the original simple linear deduction system of Loveland [10], Luckham [12], and Zamov and Sharonov [15], is actually more powerful in that more lemmas are available for use in ancestor replacement (see below).

Most linear deduction systems allow the use of lemmas: that is, any clause which has been deduced in the course of the current proof may be used as an input clause, as though it belongs to the set S, whose unsatisfiability the system is trying to establish. The linear structure of these systems, however, precludes the use of many lemmas which are available in the construction of plans.

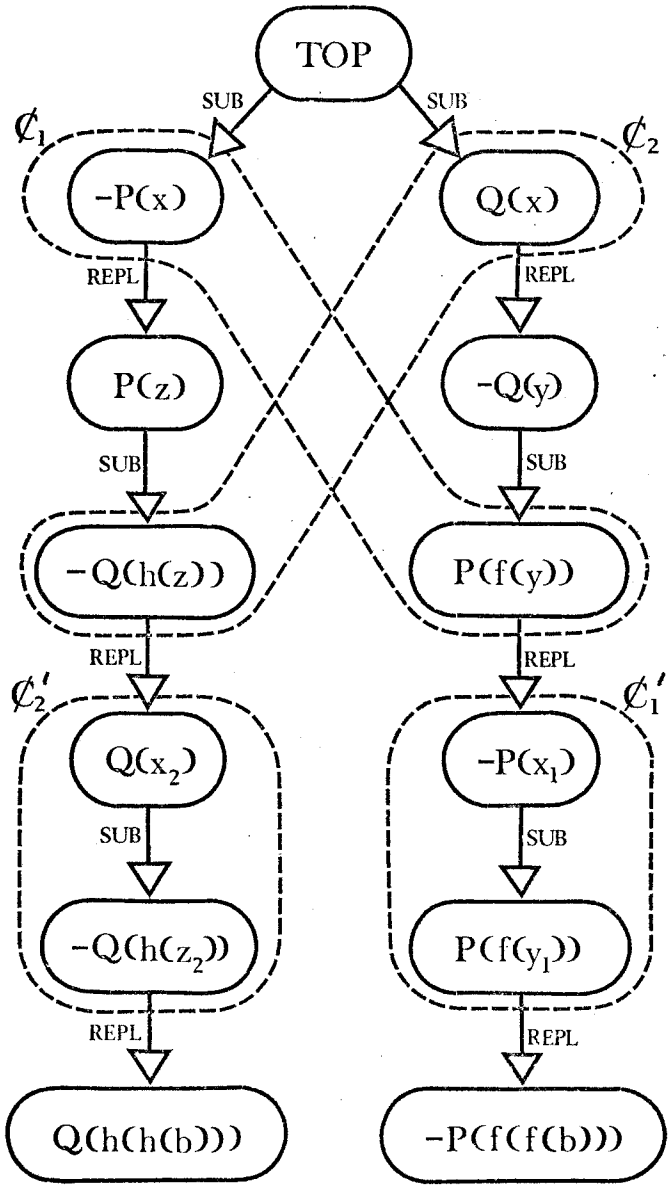
5.2.1: Example: Let S be the set of clauses:

$$\{\{-P(x), Q(x)\},$$
$$\{-Q(y), P(f(y))\},$$
$$\{P(z), -Q(h(z))\},$$
$$\{-P(f(f(b)))\},$$
$$\{Q(h(h(b)))\}\ \}$$

where b is a constant. Figure 5.1 illustrates a closed, correct plan for S the construction of which requires two ancestor replacements using variants C_1' and C_2' of clause C_1 and C_2 deduced by subplans as indicated in the diagram. In a linear system, once $-P(x)$ in the top clause is closed, it is no longer available for use in a lemma; similarly for $Q(x)$. One of these subproblems has to be solved first, however, so that only one of the two lemmas C_1 and C_2 used in generating the plan is available.

If a deduction system is to have access to the variety of lemmas which are available in plan construction, each literal used in a proof in that deduction system must be represented at least once. In a plan, each literal is represented exactly once, so among systems which use ancestor replacement, ours attains the best possible economy of representation.

There have been other attempts at representational economy in theorem-proving programs. Boyer and Moore in [4], suggested a method for representing resolvents of clauses by a system of pointers to parent clauses, and to resolved literals. In their system, as in ours, each literal is represented only once: theirs, however, is strictly a method of representation, and solves none of the problems associated with efficient backtracking, use of



A plan for the set of clauses of example 5.2.1.

Figure 5.1

lemmas, ordering of subgoals, etc. Although clauses are not explicitly created, they exist implicitly; also, substitutions are performed implicitly. Therefore, in order to perform a resolution, it is necessary to search recursively through the structure to carry out the unification and implicit construction of the resolvent.

The use of unification is also more economical in plans than in other deduction systems, since the unification algorithm is used only to verify the applicability of the rules: whenever a plan is closed, we have a refutation provided that the constraint set is unifiable. Substitutions are therefore never performed, and mgus are not calculated. In this regard, our system is similar to Huet's higher-order constrained resolution system [6].

A major difficulty with using problem-reduction in predicate calculus is that the subproblems are usually not independent. In solving a particular subproblem, we may destroy our chances of finding a solution to another subproblem. To take advantage of the problem-reduction method, therefore, we must process the subproblems in essentially a breadth-first fashion, so that if the attempted solution of subproblem A blocks the solution of subproblem B, then this fact is discovered as soon as possible, before great effort is expended on a solution for A that must eventually be erased.

Ordinary linear deduction with factoring and ancestor resolution [10,12,15] allows subproblems to be solved in any order, but lacks the power of plans in that it has no reduction rule, and its use of lemmas is comparatively restricted. When reduction is used in a linear format [7, 8, 9, 11], a strict ordering must be imposed on the solution of subproblems

to ensure completeness. Hence, the processing is done in a depth-first manner, and the system suffers from the shortcomings mentioned above.

The open subproblems of a plan, however, may be closed in any order. So, although plans share with simple linear deduction the advantages of parallel processing, they are also more powerful than the more sophisticated linear deduction systems.

5.3: Backtracking

An important consideration when deciding how constraints are to be processed, involves backtracking: a problem which to date has received little or no attention from researchers in the field of mechanical deduction.

At each point in the search for a proof, there is usually a variety of possible actions which can be performed by a theorem-proving system: it must choose the subproblem to work on next, then choose which of several solutions to that subproblem it should try. If the system should fail to solve a subproblem, it must return to an earlier point in the search, and attempt an alternative solution to an earlier subproblem. This action is termed "backtracking". The usual strategy employed in backtracking, is to return to the last point in the search at which there exists an untried alternative solution. The wastefulness of this exhaustive approach is illustrated by the following example.

5.3.1: Example: Let S be the set of clauses:

(1) Q(x,y), P(x), P(a)

(2) -P(x)

(3) -Q(x,y), S(y), R(x,y)

(4) -R(x,x), -Q(y,z), P(x)

(5) -S(x), T(x)

(6) -T(x), M(x), M(y)

(7) -M(b)

where a and b are constants.

We present a deduction from S using model elimination with factoring [8] in which subproblems are solved from right to left and the rules are applied in the order contraction, factoring, reduction and extension; and input clauses for extension are selected in the above order.

In the following search, A-literals are framed, and the rules applied are recorded to the right in abbreviated form: for example "ext(1)" means extension using clause (1).

(1)	Q(x,y), P(x), P(a)	top
(8)	Q(a,y), P(a)	fact
(9)	Q(a,y), $\boxed{P(a)}$	ext(2)
(10)	Q(a,y)	cont
(11)	Q(a,y), S(y), R(a,y)	ext(3)
(12)	$\boxed{Q(a,a)}$, S(a), $\boxed{R(a,a)}$, -Q(w,z), P(a)	ext(4)
(13)	$\boxed{Q(a,a)}$, S(a), $\boxed{R(a,a)}$, -Q(w,z), $\boxed{P(a)}$	ext(2)
(14)	$\boxed{Q(a,a)}$, S(a), $\boxed{R(a,a)}$, -Q(w,z)	cont
(15)	$\boxed{Q(a,a)}$, S(a), $\boxed{R(a,a)}$	red
(16)	$\boxed{Q(a,a)}$, S(a)	cont

(17) [Q(a,a)], [S(a)], T(a) ext(5)

(18) [Q(a,a)], [S(a)], [T(a)], M(a), M(y) ext(6)

(19) [Q(a,a)], [S(a)], [T(a)], M(a) fact

backtrack to (18)

(20) [Q(a,a)], [S(a)], [T(a)], M(a), [M(b)] ext(7)

(21) [Q(a,a)], [S(a)], [T(a)], M(a) cont

backtrack to (14)

(22) [Q(a,a)], S(a), [R(a,a)], [-Q(w,z)], P(w), P(a) ext(1)

(23) [Q(a,a)], S(a), [R(a,a)], [-Q(a,z)], P(a) fact

(24) [Q(a,a)], S(a), [R(a,a)], [-Q(a,z)], [P(a)] ext(2)

(25) [Q(a,a)], S(a), [R(a,a)], [-Q(a,z)] cont

(26) = (15) cont

(27) = (16) cont

(28) = (17) ext(5)

(29) = (18) ext(6)

(30) = (19) fact

backtrack to (29)

(31) = (20) ext(7)

(32) = (21) cont

backtrack to (22)

(33) [Q(a,a)], S(a), [R(a,a)], [-Q(w,z)], P(w), [P(a)] ext(2)

(34) [Q(a,a)], S(a), [R(a,a)], [-Q(w,z)], P(w) cont

(35) [Q(a,a)], S(a), [R(a,a)], [-Q(w,z)], [P(w)] ext(2)

(36) [Q(a,a)], S(a), [R(a,a)], [-Q(w,z)] cont

(37) = (15) cont

(38) = (16) cont

(39) = (17) ext(5)

(40) = (18) ext(6)

(41)	≠ (19)	fact
	backtrack to (40)	
(42)	= (20)	ext(7)
(43)	= (21)	cont
	backtrack to (1)	
(44)	Q(x,y), P(x), **P(a)**	ext(2)
(45)	Q(x,y), P(x),	cont
(46)	Q(x,y), **P(x)**	ext(2)
(47)	Q(x,y)	cont
(48)	**Q(x,y)**, S(y), R(x,y)	ext(3)
(49)	**Q(x,x)**, S(x), **R(x,x)**, -Q(w,z), P(x)	ext(4)
(50)	**Q(x,x)**, S(x), **R(x,x)**, -Q(w,z), **P(x)**	ext(2)
(51)	**Q(x,x)**, S(x), **R(x,x)**, -Q(w,z)	cont
(52)	**Q(x,x)**, S(x), **R(x,x)**	red
(53)	**Q(x,x)**, S(x)	cont
(54)	**Q(x,x)**, **S(x)**, T(x)	ext(5)
(55)	**Q(x,x)**, **S(x)**, **T(x)**, M(x), M(y)	ext(6)
(56)	**Q(x,x)**, **S(x)**, **T(x)**, M(x)	red
(57)	**Q(b,b)**, **S(b)**, **T(b)**, **M(b)**	ext(7)
(58)	**Q(b,b)**, **S(b)**, **T(b)**	cont
(59)	**Q(b,b)**, **S(b)**	cont
(60)	**Q(b,b)**	cont
(61)	□	cont

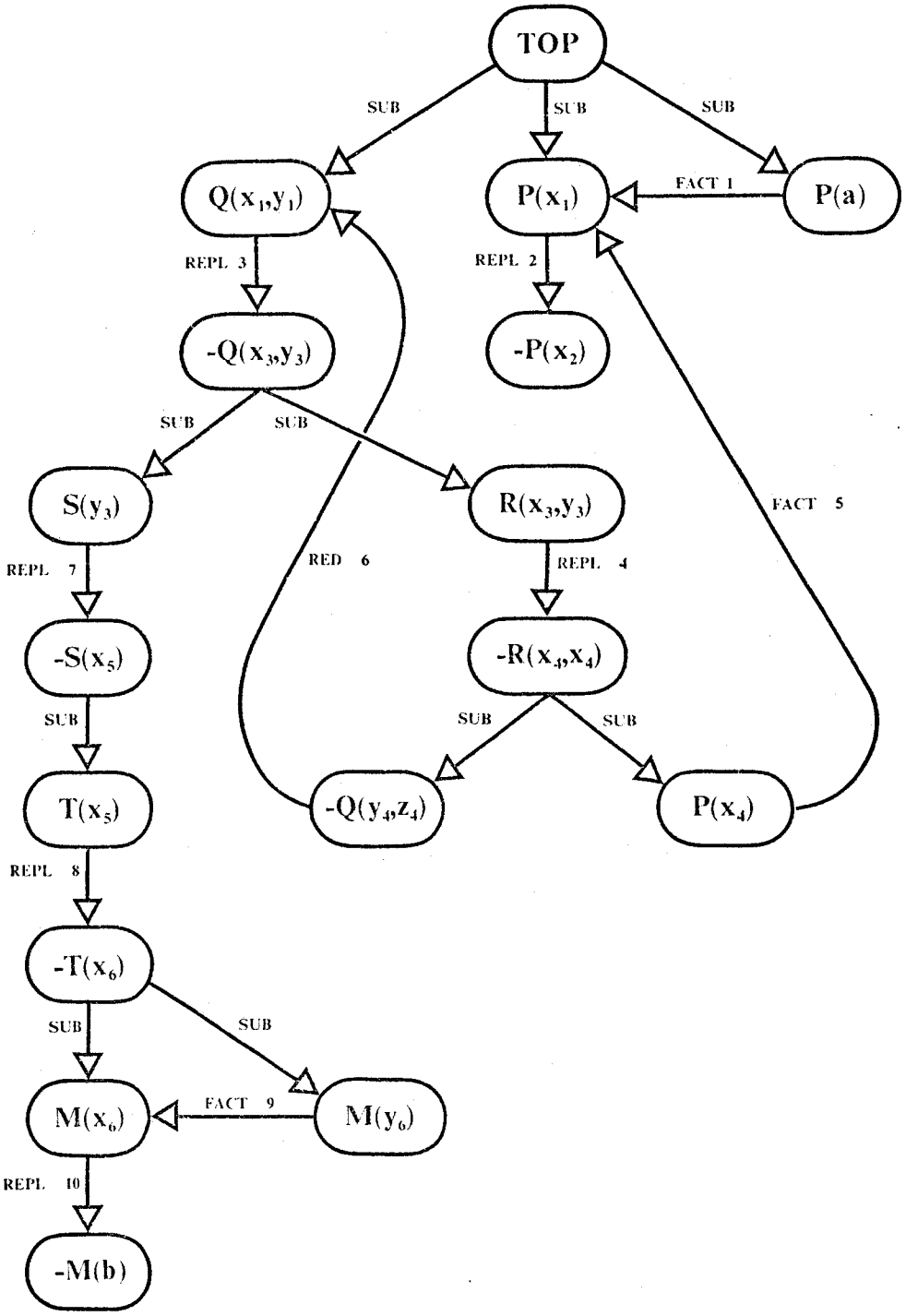
Six backtrackings are performed before the source of nonunifiability is discovered. Note also that when the correct cutting point is finally discovered at clause (43), in the above deduction, all previously found subproofs are lost even though they are correct, and are reproduced in clauses (45) to (56). In fact, between the various backtrackings, parts of the proof are generated several times: for instance, the subproblem M(x) corresponding to the third literal of clause (6) in S, is closed seven times.

Plans do not suffer from the deficiencies of linear deduction which are illustrated by the above example. Firstly, when an unsuccessful unification is attempted, it is possible to detect all sources of this nonunifiability by analysing the set of constraints. By associating with each constraint information concerning its origin, we can then determine the largest subplans which have unifiable constraint sets, then "backtrack" to one of these. Since no substitutions were applied to the vertices, no alterations need to be made to the remaining part of the plan. The reader should note that this approach to backtracking sheds an entirely new light on the proof-construction process, since at any time during the search, the history of the plan-construction can be reviewed in every detail and altered if necessary: this is not possible in other systems since some vital information is always lost. Clearly although all linear deduction strategies are applicable, new strategies need to be developed which take advantage of all the information available in plans. Details of constraint processing and backtracking can be found in [5] and future publications. Secondly, when a plan is pruned after backtracking the only parts removed are those strictly involved with the

nonunifiability: this is not the case with linear deduction. The following example, using the same set of clauses as example 5.3.1, illustrates these points.

5.3.2: Example: Figure 5.2 illustrates a closed plan G for the set of clauses of example 5.3.1. The construction of the plan simulates the linear deduction performed in that example. The constraint set for this plan is shown in figure 5.3: the reader may easily verify that C(G) is not unifiable. Using the process described in [5], we discover that by removing the arc 1, we obtain a correct subplan which is not a subgraph of any larger correct subplan. This process gives all subplans with this property of "maximal correctness": in this case there are three such subplans: the other two are obtained by removing arcs 4, 5 and 6, or arc 10 respectively. Since $M(x_6)$ has no solution other than that represented by the arc 10, we will not backtrack by removing this arc. This leaves two choices: remove arcs 4, 5 and 6, or remove arc 1. If the strategy employed is to remove as little as possible, we would remove arc 1. There is then only one choice for closing P(a); that is, by replacement using the clause {-P(x)}.

As the reader has probably noticed, backtracking to one of the maximal correct subplans could result in the system eventually generating a graph which is not a plan, since not all subplans are plans. This will not cause unsoundness, however, in view of lemma 4.1.2.



A closed plan G for the set of clauses of example 5.3.1. The integer labels on the arcs of SOL(G) indicate the order of construction of G.

Figure 5.2

REFERENCES

1. Baxter, L.D., A practically linear unification algorithm. Research Report CS-76-13, Department of Computer Science, University of Waterloo, 1976.

2. Baxter, L.D., The complexity of unification. Ph.D. Thesis, Department of Computer Science, University of Waterloo, 1976.

3. Bondy, J.A., and Murty, U.S.R., Graph Theory with Applications, American Elsevier, New York, 1977.

4. Boyer, R.S., and Moore, J.S., The sharing of structure in theorem-proving programs. Machine Intelligence 7, John Wiley and Sons, New York, 1972, pp. 101-116.

5. Cox, P.T., Deduction plans: a graphical proof procedure for the first-order predicate calculus. Ph.D. Thesis, Department of Computer Science, University of Waterloo, 1977.

6. Huet, G.P., Constrained resolution: a complete method for higher order logic. Report 1117, Jennings Computing Center, Case Western Reserve University, 1972.

7. Kowalski, R.A., and Kuehner, D., Linear resolution with selection function. Artificial Intelligence 2, (1971), pp. 227-260.

8. Loveland, D.W., Mechanical theorem proving by model elimination, J.ACM 15 (April, 1968), pp. 236-257.

9. Loveland, D.W., A simplified format for the model elimination theorem proving procedure. J.ACM 16 (July, 1969), pp. 349-363..

10. Loveland, D.W., A linear format for resolution. Lecture Notes in Mathematics 125 (Symposium on Automatic Demonstration), Springer-Verlag, Berlin, 1970, pp. 147-162.

11. Loveland, D.W., A unifying view of some linear Herbrand procedures. J.ACM 19 (April, 1972), pp. 366-384.

12. Luckham, D., Refinement theorems in resolution theory. Lecture notes in Mathematics 125 (Symposium on Automatic Demonstration), Springer-Verlag, Berlin, 1970, pp. 163-190.

13. Paterson, M.S., and Wegman, M.N., Linear unification. Proc. of Eighth Annual ACM symp. on Theory of Computing, 1976, pp. 181-186.

14. Robinson, J.A., A machine-oriented logic based on the resolution principle. J.ACM 12 (Jan., 1965), pp. 23-41.

15. Zamov, N.K., and Sharonov, V.I., On a class of strategies which can be used to establish decidability by the resolution principle. Issled po konstruktivnoye matematikye i matematicheskoie logikye III 16 (1969), pp. 54-64.