March 1970

RESOLUTION GRAPHS

by

Robert A. Yates
Bertram Raphael
Stanford Research Institute
Menlo Park, California

Timothy P. Hart*
Air Force Cambridge Research Laboratories
L. G. Hanscom Field
Bedford, Massachusetts

Artificial Intelligence Group

Technical Note 24

SRI Project 8259

*Presently at Evans, Griffiths and Hart, Inc.
Lexington, Massachusetts

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAR 1970		**00-00-1970 to 00-00-1970**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Resolution Graphs	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **SRI International,333 Ravenswood Avenue,Menlo Park,CA,94025**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper introduces a new notation, called "resolution graphs, 11 for deductions by resolution in first-order predicate calculus. A resolution graph consists of groups of nodes that represent initial clauses of a deduction and links that represent unifying substitutions. Each such graph uniquely represents a resultant clause that can be deduced by certain alternative but equivalent sequences of resolution and factoring operations. Resolution graphs are used to illustrate the significance of merges and tautologies in proofs by resolution. Finally, they provide a basis for proving the completeness of a proof strategy that combines the set of support, resolution with merging, linear format, and Loveland's subsumption conditions.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**61**	

RESOLUTION GRAPHS

Abstract

by

Robert A. Yates
Bertram Raphael
Timothy P. Hart

This paper introduces a new notation, called "resolution graphs," for deductions by resolution in first-order predicate calculus. A resolution graph consists of groups of nodes that represent initial clauses of a deduction and links that represent unifying substitutions. Each such graph uniquely represents a resultant clause that can be deduced by certain alternative but equivalent sequences of resolution and factoring operations.

Resolution graphs are used to illustrate the significance of merges and tautologies in proofs by resolution. Finally, they provide a basis for proving the completeness of a proof strategy that combines the set of support, resolution with merging, linear format, and Loveland's subsumption conditions.

I INTRODUCTION

Automatic theorem proving in the first-order predicate calculus has become an active and fruitful field for research, particularly since Robinson's landmark paper (1965)* introduced the resolution rule of inference. Computer programs that perform logical deduction by using some variation of resolution have been applied to a variety of problem domains, including lattice theory (Guard, et al., 1969), question-answering systems (Green and Raphael, 1968), and problem-solving tasks (Green, 1969).

The major reason these programs have had only limited effectiveness is that they require excessive amounts of computer time and space. This is largely due to the weakness of existing strategies for deciding how to apply the resolution rule. Robinson presented one logically complete (but grossly inefficient) strategy for using resolution to prove theorems. Since then numerous papers have appeared describing more restrictive strategies that are also complete. The nature of some of these strategies is syntactic, i.e., they depend upon the identity or number of elementary symbols (Wos et al., 1964; Wos et al., 1965; Andrews, 1968; Loveland, 1970). Other strategies are of a semantic nature, i.e. they depend upon assignments of models or truth values (Slagle, 1967; Luckham, 1968).

In this paper we prove the completeness of an extremely restrictive syntactic strategy. Raphael (1969) presented part of this strategy in

---

*References appear at the end of this paper.

an informal note a year ago. (This paper supercedes that note.) Since then, its completeness has been independently established by Anderson and Bledsoe (1970).

The completeness of this strategy was discovered by us while working with a new graphical notation for resolution deductions. This notation, whose justification is based upon the mathematical concept of a partition, leads us to a clearer understanding and simplified proofs for several existing theorems in resolution theory. Therefore a major purpose of this paper is to present the idea of a resolution graph and show its usefulness.

II TERMINOLOGY

We shall assume familiarity with standard terminology and notation of first-order predicate calculus and proof by resolution. In summary, all logical statements are assumed to be in quantifier-free conjunctive-normal form; existential quantifiers are eliminated by the introduction of Skolem functions, and universal quantification is assumed over variables. The initial information is represented by a finite set of clauses, each of which is a set of literals. (This finite set is called the "clause form" of the predicate-calculus statement obtained by forming the disjunction of the literals in each clause, and then the conjunction of the resulting formulas). Each literal is either an atomic formula or the negation of an atomic formula. An atomic formula consists of a predicate symbol and an appropriate number of terms for its arguments.

Each term is either a constant, a variable, or the composition of a function applied to an appropriate number of terms as arguments.

Resolution may be taken to be an operation mapping two parent clauses B and C into a "resolvent" clause D. By Robinson's definition, the clause D is a resolvent of B and C (which have been "standardized" to have no variable names in common) iff (if and only if) there are nonempty subsets $b \subseteq B$ and $c \subseteq C$ such that the atomic formulas in $b \cup c$ are unifiable with a most general unifier σ, $b\sigma = \{L_1\}$, $c\sigma = \{L_2\}$, where L_1 and L_2 are complementary literals, and $D = (B\sigma - \{L_1\}) \cup (C\sigma - \{L_2\})$. We call elements $b \cup c$ the literals resolved upon. (The concepts of unifier and most general unifier will be discussed in detail in the next section of the paper.)

Andrews (1968) broadens this definition of resolvent by not requiring σ to be most general. This is equivalent to considering every substitution instance of the above D also to be a resolvent of B and C.

A common restriction of resolution is simple resolution, in which b and c must be singletons. It can be shown that an inference system based upon simple resolution alone is not complete.

If some clause D has a subset d and σ is a most-general unifier of the literals in d, $d\sigma = \{L\}$, then we call the clause $D\sigma$ a factor of D, and D a parent of its factor. Since a clause implies all of its instances, clearly a clause implies all of its factors. One can show that simple resolution and factoring can form a complete inference system. [Although factoring played a major role in an early unpublished version of Robinson (1965)

and was implemented in the first resolution program (Wos et al. (1964, 1965)), it has been little discussed in the literature. A variation called distinguished literal factoring is discussed by Kowalski and Hayes (1969)]. In fact, resolution in Robinson's sense may be viewed as a simple resolution that has been preceded, if necessary, by appropriate factoring operations on the parents. In this paper we shall generally view a resolution inference step as consisting of the two phases: factoring followed by simple resolution.

A deduction of clause C from an initial set of clauses S is a finite sequence of clauses B_1, B_2, ..., B_n such that:

(1) B_i, $1 \leq i \leq n$, is either in S or it is a resolvent of B_j and B_k, $1 \leq j, k < i$. and

(2) $B_n = C$.

A refutation of a set S of clauses is a deduction from S of the empty clause, which is denoted by $\square$. The usual way to attempt to verify that a theorem T is deducible from a set of axiomatic clauses $\mathcal{G}$ is to attempt to construct a refutation of the set $\mathcal{G} \cup \overline{\mathcal{T}}$, where $\overline{\mathcal{T}}$ is the clause form of the negation of the theorem T.

If D is either a resolvent or a factor, each literal $L \in D$ is equal to $L'\sigma$ for at least one literal L' in a parent clause (and the appropriate substitution σ). Every such L' is called a parent of literal L. Thus in any deduction of a clause D from a set S of initial clauses, we may trace the ancestry of the literals in D back to literals in the members of S.

Andrews (1968) defines a merge to be any deduced clause containing a literal that has parent literals in both parent clauses. We find it

convenient to extend this concept by defining a d-merge ("descent merge") to be any clause containing any literal L that has ancestors in two or more distinct occurrences of members of S, and we call such an L a d-merge literal. Every d-merge is thus either a merge or the descendant of a merge. Since the formation of a merge always causes two or more literals in different initial clauses to become associated (by virtue of becoming ancestors of the same literal), we shall sometimes use the term "merge" merely to refer to this association, which is exhibited as an explicit link in the "resolution graphs" to be defined below. Note that L may have two or more ancestors and yet not be a d-merge literal (when all its ancestors occur in the same occurrence of a member of S).

Subsumption is an important phenomenon in most resolution proof strategies. For any two clauses C and D, C is said to subsume D if an instance of C is contained in D, i.e., there exists a substitution σ such that $C\sigma \subseteq D$. If D is subsumed by C, it is implied by C and generally may be replaced by C in the construction of a deduction. Note that a tautology (a clause containing a pair of complementary literals) cannot subsume any nontautological clause, since substitution cannot destroy the tautologousness.

III PARTITIONS AND UNIFICATION

We shall now explore some of the properties of the unification operation. The goal of this discussion is to clarify the possible effects of composing or permuting substitutions.

Unification Theorem

Let $E = \{e_1, e_2, \ldots, e_n\}$ be any set of expressions (e.g., all the atomic formulas that occur in some set S of initial clauses). Let

$\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$ be a class of nonempty subsets of E (e.g., think of each E_i as containing a different set of atomic formulas from E that might be made identical by an appropriate substitution). Thus $E_i \subseteq E$ and $E_i \neq \phi$ (the empty set) for all i. If θ is any substitution, then $\mathcal{E}\theta = \{E_1\theta, E_2\theta, \ldots, E_m\theta\}$, where $E_i\theta$ is the set of expressions $e_j\theta$ obtained by making the substitution defined by θ in each e_j in E_i.

A class $\mathcal{E}$ is said to be unifiable if there exists a substitution θ such that $E_i\theta$ is a singleton, i.e., contains only one element, for every i; and such a θ is said to unify $\mathcal{E}$.

The well-known unification theorem can be stated as follows:

Unification Theorem--Let $\mathcal{E}$ be a unifiable class of subsets of a set E of expressions. Then there exists a most general unifier $\sigma_{\mathcal{E}}$ of $\mathcal{E}$ with the property that for any unifier θ of $\mathcal{E}$, there is a substitution λ such that θ is the composition $\sigma_{\mathcal{E}} \circ \lambda$ of substitutions $\sigma_{\mathcal{E}}$ followed by λ. Thus every unifier of $\mathcal{E}$ is an instance of the most general unifier of $\mathcal{E}$. (A proof and discussion of the Unification Theorem appears in Robinson (1967).)

Clearly, "the" most general unifier of a class is not unique; any most general unifier of $\mathcal{E}$, when composed with any invertible substitution, is again a most general unifier. For example, the class

$$\mathcal{E} = \{\{x, g(y)\}, \{x, g(u)\}\}$$

has both $\theta_1 = [g(y)/x, y/u]$ and $\theta_2 = [g(u)/x, u/y]$ as most general unifiers. However, θ_2 can be obtained from θ_1 by the invertible substitution $[y/u, u/y]$. In fact, it is a corollary of the Unification Theorem that this is the only way most general unifiers of a set can be related: If $\sigma_{\mathcal{E}}$ and $\sigma'_{\mathcal{E}}$ are most general unifiers of $\mathcal{E}$ they are instances of one another and consequently are alphabetic variants of each other. Since all such $\sigma_{\mathcal{E}}$ are essentially equivalent, henceforth we shall assume that $\sigma_{\mathcal{E}}$ is unique.

Partitions

Let $\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$ and $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ be two classes of nonempty subsets of a set E of expressions (e.g., if E contains all the atomic formulas in some set S of initial clauses, let the E_i be sets of potentially unifiable atomic formulas from some subset of S, and let the F_i be the corresponding sets for a different, perhaps larger, subset of S.)

We define the following partial ordering on the classes of subsets of E: $\mathcal{E} \leq \mathcal{F}$ iff, for each i, $E_i \subseteq F_j$ for some j.

The class $\mathcal{E}$ is called a partition provided $E_1, \ldots, E_m$ are mutually disjoint sets. By the closure of $\mathcal{E}$, denoted by $[\mathcal{E}]$, we mean the smallest partition such that $\mathcal{E} \leq [\mathcal{E}]$. The class $[\mathcal{E}]$ is formed from $\mathcal{E}$ by successively merging together sets E_i, E_j of $\mathcal{E}$ with an element in common until all the sets so obtained are mutually disjoint.

Example: If $\mathcal{E} = \left\{ \{a\}, \{x, f(y)\}, \{u, f(y)\}, \{g(x,y), b, h(u)\}, \{f(x), h(u), c\}, \{g(f(x),y), c\} \right\}$,

then $[\mathcal{E}] = \left\{ \{a\}, \{x, f(y), u\}, \{g(x,y), b, c, f(x), h(u), g(f(x),y)\} \right\}$.

Induced Partitions

If E is a set of expressions and θ a substitution, then θ induces a partition P_θ on E defined as follows: Two expressions e_i and e_j in E lie in the same block of P_θ iff $e_i\theta = e_j\theta$. For example,

if $E = \{f(x), u, g(x,y), f(h(v))\}$

and $\theta = [h(v)/x, g(h(v),y)/u]$, then θ induces the partition

$P_\theta = \{ \{f(x), f(h(v))\}, \{u, g(x,y)\} \}$ on E.

If $\mathcal{E}$ is a unifiable class of subsets of E, and θ unifies $\mathcal{E}$, then the partition P_θ induced by θ on the expressions occurring in $\mathcal{E}$ has the property that $\mathcal{E} \leq P_\theta$, because if $E_i \in \mathcal{E}$ then all the expressions in E_i are unified by θ and hence E_i is contained in a single block of P_θ. Since the closure $[\mathcal{E}]$ of $\mathcal{E}$ is the smallest partition containing $\mathcal{E}$, we have $\mathcal{E} \leq [\mathcal{E}] \leq P_\theta$. Conversely, a class $\mathcal{E}$ is unifiable if $\mathcal{E} \leq P_\theta$ (or $[\mathcal{E}] \leq P_\theta$) for some substitution θ.

Example: $\theta = \{f(a)/x,\ a/y\}$

$$\mathcal{E} = \{\{x,\ f(y)\},\ \{x,\ f(a)\},\ \{y\},\ \{a\}\}$$

$$[\mathcal{E}] = \{\{x,\ f(y),\ f(a)\},\ \{y\},\ \{a\}\}$$

$$P_\theta = \{\{x,\ f(y),\ f(a)\},\ \{y,\ a\}\} \quad .$$

Now, if $\sigma_\mathcal{E}$ is a most general unifier of $\mathcal{E}$, then $\sigma_\mathcal{E} = \sigma_{[\mathcal{E}]}$. This is true because if two blocks of $\mathcal{E}$ contain an element e in common, any unifier θ of $\mathcal{E}$ must unify both blocks to the same element $e\theta$. Therefore, the unifier of a class of sets of expressions depends only upon the partition $[\mathcal{E}]$, the closure of the class. It also follows that if $\mathcal{E}$ and $\mathcal{F}$ are classes such that $\mathcal{F} \leq \mathcal{E}$ and $\mathcal{E}$ is unifiable, then, since $[\mathcal{E} \cup \mathcal{F}] = [\mathcal{E}]$, we have:

$$\sigma_{\mathcal{E} \cup \mathcal{F}} = \sigma_\mathcal{E} \quad .$$

Equivalent Substitutions

The following lemma shows an important invariance of unification to the order in which substitutions are performed:

Lemma 1: If $\mathcal{E}$ and $\mathcal{F}$ are unifiable classes with most general unifiers $\sigma_\mathcal{E}$ and $\sigma_\mathcal{F}$ respectively, then $\mathcal{E} \cup \mathcal{F}$ is unifiable if and only if $\sigma_{(\mathcal{F}\sigma_\mathcal{E})}$ exists (i.e., $\mathcal{F}\sigma_\mathcal{E}$ is unifiable). In this case $\sigma_{(\mathcal{E}\sigma_\mathcal{F})}$ also exists, and $\sigma_\mathcal{E} \circ \sigma_{(\mathcal{F}\sigma_\mathcal{E})}$ and $\sigma_\mathcal{F} \circ \sigma_{(\mathcal{E}\sigma_\mathcal{F})}$ are both most general unifiers of $\mathcal{E} \cup \mathcal{F}$.

Proof: We first show that if $\mathcal{E} \cup \mathcal{F}$ is unifiable, then $\sigma_{(\mathcal{F}\sigma_{\mathcal{E}})}$ exists and that $\sigma_{\mathcal{E}\cup\mathcal{F}} = \sigma_{\mathcal{E}} \circ \sigma_{(\mathcal{F}\sigma_{\mathcal{E}})}$. Let θ be any unifier of $\mathcal{E} \cup \mathcal{F}$. Since θ unifies $\mathcal{E}$, we can write $\theta = \sigma_{\mathcal{E}} \circ \lambda$ for some substitution λ. But for each $F_i \in \mathcal{F}$, $F_i\theta = (F_i\sigma_{\mathcal{E}})\lambda$ is a singleton, so λ unifies $\mathcal{F}\sigma_{\mathcal{E}}$. Consequently, $\sigma_{(\mathcal{F}\sigma_{\mathcal{E}})}$ exists, and $\lambda = \sigma_{(\mathcal{F}\sigma_{\mathcal{E}})} \circ \mu$ for some μ; so we have $\theta = \sigma_{\mathcal{E}} \circ \sigma_{(\mathcal{F}\sigma_{\mathcal{E}})} \circ \mu$. Since $\sigma_{\mathcal{E}} \circ \sigma_{(\mathcal{F}\sigma_{\mathcal{E}})}$ does in fact unify $\mathcal{E} \cup \mathcal{F}$ and since any unifier θ of $\mathcal{E} \cup \mathcal{F}$ can be factored as $\theta = \sigma_{\mathcal{E}} \circ \sigma_{(\mathcal{F}\sigma_{\mathcal{E}})} \circ \mu$ for some μ, then $\sigma_{\mathcal{E}} \circ \sigma_{(\mathcal{F}\sigma_{\mathcal{E}})}$ is in fact a most general unifier of $\mathcal{E} \cup \mathcal{F}$. Conversely, if we assume that $\sigma_{(\mathcal{F}\sigma_{\mathcal{E}})}$ does exist, then $\sigma_{\mathcal{E}} \circ \sigma_{(\mathcal{F}\sigma_{\mathcal{E}})}$ unifies $\mathcal{E} \cup \mathcal{F}$, and the above argument shows that it is its most general unifier. The result is clearly symmetric in $\mathcal{E}$ and $\mathcal{F}$.

An immediate corollary is that if $\mathcal{F} \leq \mathcal{E}$, then

$$\sigma_{\mathcal{E}} = \sigma_{\mathcal{F}} \circ \sigma_{(\mathcal{E}\sigma_{\mathcal{F}})} \quad (\text{since } \sigma_{\mathcal{E}} = \sigma_{(\mathcal{E}\cup\mathcal{F})}) \ .$$

This says that the most general unifier $\sigma_{\mathcal{E}}$ of $\mathcal{E}$ can be computed by first unifying a subclass $\mathcal{F}$ of $\mathcal{E}$ and then unifying the remainder of $\mathcal{E}$ with the unified $\mathcal{F}$. By induction we have a second corollary:

Corollary: Let $\mathcal{E} = \{E_1, \ldots, E_n\}$ be a class of sets of expressions.

Let $\sigma_1 = \sigma_{\{E_1\}}$, $\sigma_2 = \sigma_1 \circ \sigma_{\{E_2\}\sigma_1}$, $\ldots$, $\sigma_n = \sigma_{n-1} \circ \sigma_{\{E_n\}\sigma_{n-1}}$.

Then $\sigma_{\mathcal{E}}$ exists if and only if $\sigma_1, \ldots, \sigma_n$ all exist, and $\sigma_{\mathcal{E}} = \sigma_n$.

Proof: For $n = 1$ there is nothing to prove.

For $n > 1$ let $\mathcal{F} = \{E_1, \ldots, E_{n-1}\}$. By the induction hypothesis,

$\sigma_{\mathcal{F}}$ exists iff $\sigma_1, \ldots, \sigma_{n-1}$ exist and $\sigma_{\mathcal{F}} = \sigma_{n-1}$. So $\sigma_{\mathcal{E}} = \sigma_{\mathcal{F}} \circ \sigma_{\{E_n\}\sigma_{\mathcal{F}}} = \sigma_n$ and $\sigma_{\mathcal{E}}$ exists iff $\sigma_{\{E_n\}\sigma_{\mathcal{F}}}$ exists i.e., iff $\sigma_1, \ldots, \sigma_n$ all exist.

Since $\sigma_{\mathcal{E}}$ is independent of the sequence numbering of $E_1, \ldots, E_n$, it is clearly independent of the sequence of operations used to compute it.

Example: Let $\mathcal{E} = \left\{\left\{x, f(u, g(v)), f(a,y)\right\}, \left\{g(b), y\right\}\right\}$

$$\mathcal{F} = \left\{\left\{f(u, g(v)), f(a,y)\right\}\right\} \qquad \text{so } \mathcal{F} \leq \mathcal{E}$$

then $\sigma_{\mathcal{F}} = [a/u, g(v)/y]$, $\mathcal{E}\sigma_{\mathcal{F}} = \left\{\left\{x, f(a,g(v))\right\}, \left\{g(b), g(v)\right\}\right\}$

$$\sigma_{(\mathcal{E}\sigma_{\mathcal{F}})} = [f(a,g(b))/x, b/v]$$

$$\sigma_{\mathcal{E}} = \sigma_{\mathcal{F}} \circ \sigma_{(\mathcal{E}\sigma_{\mathcal{F}})} = [a/u, g(b)/y, f(a,g(b))/x, b/v] \quad .$$

IV RESOLUTION GRAPHS

One of the main results of this paper is the fact, to be stated formally in Theorem 1, that certain alternative sequences of simple resolution and factoring operations produce precisely the same resultant clause. The purpose of the graphical notation developed in this section is to represent uniquely certain equivalent sets of deductions.

Let S be a set of initial clauses, and let $C = \{L_1, L_2, \ldots, L_n\}$ be a clause in S where $L_1, \ldots, L_n$ are the literals of C. We represent C as a graph by associating a circle to each literal of C and by connecting the circles by horizontal bars (Figure 1a). Such a graph is called an initial graph since it represents an initial clause. The circles of the graph are named by their corresponding literals $L_1, \ldots, L_n$

and can occur in any order; two graphs that differ only because the literals within a clause are differently sequenced will be considered equivalent. These initial graphs are the building blocks; more complicated graphs are always constructed from these by operations corresponding to resolution and factoring. Before stating a formal definition of the graph structures we call resolution graphs, we first give some instances of how these graphs are formed and introduce some additional terminology.

If $D = \{M_1, \ldots, M_\ell\}$ is another initial clause which is resolved against C on the literals L_i and M_j, we represent the resolvent clause R(C,D) as a graph by connecting the two corresponding circles (in the initial graphs for C and D) by a double bar (Figure 1b). The literals of the clause R(C,D) are the names of the circles in the new graph which are not connected by double bars. These literals (marked by primes in Figure 1b) are assumed to be those of the initial clauses instantiated by the most general unifier of the atomic formulas of L_i and M_j (which reduced L_i and M_j to the complementary literals L_i' and M_j'). These instantiated literals are said to be associated to their corresponding circles. However, we shall continue to refer to the circle by its original literal name. Should two or more literals collapse together, i.e., become identical, as the result of a resolution operation, we indicate this in the graph by connecting the literals (circles) together by a dotted line--thereafter forcing these circles to be considered as a single node (see Figure 1c). (We define a node of a graph to be a dot-connected group of circles--or a single circle which is not dot-connected.) For example, if the most general unifier of L_i and M_j also happens to make L_1 and M_1 into identical

literals, we represent the resolvent clause by the graph shown in Figure 1c. However, we prefer to consider the graph operations of adding double bars and dotted lines as independent operations. Simple resolution applied to graphs is accomplished by a double bar operations followed by dotted line operations if the literals should collapse. In general, we allow graphs in which two distinct circles have the same associated literal without demanding that the circles be dot-connected.

The dotted line is also used to indicate an explicit factoring operation. If L_1' and M_1' in Figure 1b are distinct but unifiable, then Figure 1c represents the result of applying the factoring substitution to the resolvent clause. The associated literals L_1', ..., M_ℓ' would be replaced by their respective instances.

We say a graph node is free if it is not connected to any other node by a double bar, and a circle is free if the node to which it belongs is free. Graph operations are only performed on free nodes. As an example, let

$$C_1 = \{\overset{L_1}{P(x)}, \overset{L_2}{Q(a)}, \overset{L_3}{R(a,x)}\}$$

$$C_2 = \{\overset{L_4}{\sim Q(y)}, \overset{L_5}{R(y,b)}\}$$

$$C_3 = \{\overset{L_6}{\sim P(b)}\}$$

$$C_4 = \{\overset{L_7}{\sim R(a,z)}, \overset{L_8}{P(f(z))}\} \quad .$$

(See Figure 2a). Let R_1 be the resolvent of C_1 and C_2:

$$R_1 = \{P(x), R(a,x), R(a,b)\} \quad .$$

Let R_2 be the resolvent of R_1 and C_3:

$$R_2 = \{R(a,b)\} \quad ,$$

and let R_3 be the resolvent of R_2 and C_4:

$$R_3 = \{P(f(b))\} \quad .$$

(Figure 2b)

We have labeled the graph operations α, β, γ, δ as seen in Figure 2b where γ is the operation of adding the dotted line. So the sequence of graph operations leading to R_3 is $\langle\beta,\alpha,\gamma,\delta\rangle$. However, we could obtain the same result by performing the operations in any one of the following orders:

(1) $\langle\beta,\alpha,\gamma,\delta\rangle$

(2) $\langle\alpha,\beta,\gamma,\delta\rangle$

(3) $\langle\beta,\gamma,\alpha,\delta\rangle$

(4) $\langle\beta,\gamma,\delta,\alpha\rangle$.

For example, the sequence $\langle\beta,\gamma,\delta,\alpha\rangle$ corresponds to first resolving C_1 and C_2 then factoring L_3 and L_5, then resolving with C_4 and finally with C_3.

It is a consequence of the corollary to lemma 1 that any two sequences of graph operations which lead to the same final graph structure produce the same resulting clause.

In Figure 2, the literals in parentheses are the associated literals; each associated literal is associated to a graph node and is an instance of one of the original literals naming a circle of the node. Since we shall show that these literals do not depend on the order of graph operations, we shall subsequently drop these literals from the diagrams.

The term "merge in a graph" will always refer to a dotted link between literals in distinct initial clauses. The resultant clause of a graph containing such a merge link is itself either a merge of a d-merge, but we shall not always be able to tell which (and the distinction will be unimportant in our development).

Graph Structure

In the preceding discussion we described the resolution graphs corresponding to certain deductions. We now give a recursive definition for the class of structures called resolution graphs or simply graphs, in the remainder of the paper. (Note: we assume in the following that every occurrence of an initial clause contains a unique set of variable symbols, so that no naming conflicts arise during the process of forming graphs.)

Definition of Graph

(1) The representation for any initial clause by a connected row of circles is a graph, and is called an initial graph.

(2) If G is a graph, and two associated literals from distinct free nodes may be made identical by the most general unifier λ, the result of connecting together all the circles in the two nodes by dotted lines is a graph. The new connected group is a single free node of the new graph and each associated literal is replaced by its λ-instance.

(3) If G_1 and G_2 are graphs and the associated literals of a free node N_1 of G_1 and a free node N_2 of G_2 may be unified, the result of dot connecting the circles of N_1 and N_2 is a graph G. Again, the new dot-connected group is a single node

of G and the associated literals are replaced by their respective instances (under the unifying substitution).

(4) If G_1 and G_2 are graphs, and if the associated literals L_1 of a free node N_1 of G_1 and L_2 of a free node N_2 of G_2 may be made complementary (i.e., identical atomic formulas and opposite signs) by an appropriate unifier λ, then the result of connecting N_1 and N_2 by a double bar is a graph G. N_1 and N_2 are no longer free in G and the associated literals of the nodes in G are λ-instances of the associated literals of G_1 and G_2.

(5) Only those structures which can be built up by a finite number of applications of rules 1, 2, 3 and 4 to some set of initial clauses are graphs.

Examples: Figure 3 shows some possible and impossible graph structures.

Definition: A set of literals (or circles) in a graph is said to be unifier-connected if the literals are connected by double bars or dotted lines or both; e.g., the graph in Figure 4a contains four unifier-connected sets:

$\{L_1, L_7\}$, $\{L_4, L_{10}\}$, $\{L_5, L_{11}\}$, and $\{L_2, L_3, L_8, L_9\}$.

Definition: Let $\mathcal{E}_G$ be the class of unifier-connected sets of literals in a graph G, except that all negation signs are omitted, so that $\mathcal{E}_G$ is a class of sets of atomic formulas. The most general unifier of the graph, denoted by σ_G, is defined to be the most general unifier of $\mathcal{E}_G$.

In the graph construction, the sets in $\mathcal{E}_G$ are successively unified; hence, by the corollary to lemma 1, σ_G always exists (assuming G exists) and the associated literals of the nodes of G are given by applying σ_G to the original literals of G. This argument proves that a resolution graph is independent of the sequence of operations by which it is constructed. The graph partition, denoted by P_G, is the partition induced by σ_G on the set of atomic formulas occurring in G. Since σ_G may unify atomic formulas which are not in $\mathcal{E}_G$, we always have:

(1) $\mathcal{E}_G \leq [\mathcal{E}_G] \leq P_G$ and

(2) $\sigma_G = \sigma_{\mathcal{E}_G} = \sigma_{[\mathcal{E}_G]} = \sigma_{P_G}$.

Repetitions

In resolution theorem proving, one generally considers the set S of initial clauses to be of fixed size, even though a single member of S may be used several times in a proof and each such use requires a new alphabetic variant of the clause. In working with graphs, we find it more convenient to consider each occurrence of an initial clause, represented by a row of circles, to be distinct. Therefore, the "set of initial clauses" of a graph--or the "set of initial graphs" may contain repetitions of members of the "set of initial clauses" in the predicate calculus sense. Similarly, when discussing the "literals" of a graph we consider each node in the structure to be associated with a distinct literal (its associated literal) even though the literals represented by two or more of them may be syntactically identical. With these ideas in mind, we find the concepts of g-clause ("graph-clause") and subgraph useful.

Definition: A g-clause is an unordered tuple of literals. Two g-clauses are equal if every member of one is a member of the other, and occurs with the same multiplicity.

Definition: The resultant of a graph G is the g-clause of associated literals of the free nodes of G. The resultant may be constructed by first forming a g-clause of initial literals, one from each free node of G, and then instantiating each member by σ_G. Each g-clause determines a unique set of literals (deleting multiple occurrences) and hence the resultant determines a unique clause called the resolvent of the graph.

Definition: A graph G′ is said to be a subgraph of another graph G provided G is constructable from G′ by applications of rules 1, 2, 3 and 4 defining graph constructions. We also call G an extension of G′.

We note that the relation "G_1 is a subgraph of G_2" is clearly transitive, and that every initial graph in a graph G is a subgraph of G.

Two subgraphs G_1 and G_2 of a graph G are said to be disjoint if they have no initial graphs in common. If G_1 and G_2 are disjoint they can have at most one double bar link connecting them. (There may be one or more dotted links between them, however). Figure 4(b) shows the 12 possible subgraphs of the graph in Figure 4(a). (Note that certain graphs may in a sense be contained within a graph and yet not be subgraphs of it as defined here. For example, the graph formed by dot-connecting L_2 and L_8 of subgraphs (1) and (2) of Fig. 4b cannot be constructively extended to the full graph of Fig. 4a--or even to subgraph (6).)

Another interesting property of subgraphs is that they can always be replaced by their resultants. Thus, if G_1 is a subgraph of G and if C_1 is the resultant of G_1, we can construct an initial graph for C_1 and replace G_1 by this new graph in G without affecting the resultant of G. This is possible since each graph operation used in constructing G from G_1 affects only the free nodes of G_1, and their associated literals. However, each such free node and corresponding literal is represented uniquely in the graph of C_1.

Deductions

As mentioned earlier, the introduction of a dotted line within a graph corresponds to a factoring operation, and the introduction of a double bar between two graphs corresponds to simple resolution. Because of the distinction between g-clauses and ordinary clauses, and because the double bar operation does not account for literal collapses, these correspondences are not complete. However, we can define simple resolution and factoring in terms of the graph operations and consequently obtain a precise correspondence between deductions and resolution graphs (or an appropriate subset thereof).

Definition: Let B_1, ..., B_n be a deduction of a clause C (B_n = C) from a set S of initial clauses. We define the graph generated by the deduction as follows:

(1) Each occurrence of an initial clause of S in the deduction is represented by a separate initial graph.

(2) For each factoring step, we apply the factoring substitution to the free associated literals and dot-connect those nodes whose associated literals have been unified by the substitution (by step 2 in the definition of a graph).

(3) For each simple resolution, the appropriate double bar is added to the graph (by step 4). Again, if any associated literals from distinct free nodes have been unified, we dot-connect these nodes.

Thus, each clause B_i in the deduction is represented by a unique graph G_i whose resultant (or resolvent in this case) is B_i (or a variant). A graph that can be generated in this way by some deduction is called a deducible graph.

Dominance

Definition: A resolution graph G_1 dominates another graph G_2 (written $G_1 \geq G_2$) provided the resolvent of G_1 subsumes the resolvent of G_2.

Using the graph structures and associated partitions we can frequently determine by inspection whether a given graph dominates another graph. A sufficient criteria for G_1 to dominate G_2 is that:

(1) The literals naming free circles of G_1 are a subset of the literals naming free circles of G_2.

(2) $P_{G_1} \leq P_{G_2}$ (alternatively $\sigma_{G_2} = \sigma_{G_1} \circ \lambda$ for some λ).

In this case $C_1\lambda \subseteq C_2$ where C_1 and C_2 are the resolvents (or resultants) of G_1 and G_2 respectively. One example is that the graph of a factored clause is dominated by the graph of its parent. A more interesting example is given as follows:

Let G_1 be the graph of Figure 5a, so $\mathcal{E}_{G_1} = \left\{ \{L_1, L_3, L_4\}, \{L_5, L_6\} \right\}$. Let $\hat{C}_1$ be a variant of C_1 and let G^* have the structure shown in Figure 5b. Then

$$\mathcal{E}_{G^*} = \left\{ \{L_1, \hat{L}_1, L_3, L_4\}, \{L_2, \hat{L}_2\}, \{L_5, L_6\} \right\} .$$

Observe that although G^* is not deducible, its resultant is identical to (or a variant of) the resultant of G_1; the variant $\hat{C}_1$ was added to show that G_2 (Figure 5c) dominates the graph G^*, because $\mathcal{E}_{G_2} = \left\{\left\{\hat{L}_1, L_3\right\}, \left\{L_2, \hat{L}_2\right\}, \left\{L_1, L_4\right\}, \left\{L_5, L_6\right\}\right\}$ and therefore $\mathcal{E}_{G_2} \leq \mathcal{E}_{G^*}$. Consequently, the resultant of G_2 subsumes the resultant of G_1. For example, suppose the initial literals in Figure 5 are:

$$L_1 = \sim P(u)$$

$$L_2 = Q(z)$$

$$L_3 = P(a)$$

$$L_4 = P(x)$$

$$L_5 = R(x)$$

$$L_6 = \sim R(y)$$

$$L_7 = S(y)$$

$$\hat{L}_1 = \sim P(\hat{u})$$

$$\hat{L}_2 = Q(\hat{z}).$$

Then the reader can verify that the resultant of G_1 or G^* is the clause $\left\{Q(z), S(a)\right\}$, while the resultant of G_2 is the stronger clause $\left\{Q(z), S(x)\right\}$. This illustrates the principle (to be proven in Lemma 2) that in general, if one resolves first and then factors, rather than vice versa, one obtains a "stronger" clause, i.e. a clause that subsumes the clause obtainable by first factoring and then resolving.

We are interested in the dominance relation for the following reason: In section V we shall deal with the construction, subject to certain constraints, of a proof represented by a certain graph G. Analogously with the above example, it turns out to be easier to construct a different but dominating graph $G' \geq G$. This is generally satisfactory since if $\mathcal{C}_{G'} \leq \mathcal{C}_G$ then for any clause that can be deduced (by simple resolution or factoring) from the resultant of G, there exists a clause at least as strong that can be deduced from the resultant of G'--as we shall show below.

Definition: A graph G is said to contain a loop if two initial graphs of G are connected by chains of unifier-connecter literals in more than one way.

Thus, in Figure 4b, the subgraphs 6, 8, 9, 10, 11 and 12 contain loops while the remainder are loop-free. We note that the only way loops can occur in deducible graphs is as a result of merges. This fact will be important in our later development of a proof strategy.

The following theorem establishes the equivalences of certain graphs, and corresponding deductions.

Theorem 1: (Resolution graph theorem). Let G be a graph representing a resultant g-clause C constructed from a set S of initial clauses (including possible repetitions), and let $\mathcal{C}_G$ be the class of unifier-connected sets of literals from G.

1. σ_G exists, and the resultant C is obtained by applying σ_G to the literals of the free nodes of G. If G is deducible, any deduction that generates G produces the same resultant clause.

2. If G is constructed from G_1 and G_2 by a double bar operation, and if G_1' dominates G_1, then either G_1' dominates G, or else G_1' and G_2 can be double bar connected producing a graph G' dominating G.

3. Let G_1 be a subgraph of G having C_1 as its resultant, and let D be any g-clause subsuming C_1. Let S_1 be the initial clauses used in G_1. Then there is a graph G' containing initial clauses from $(S - S_1) \cup \{D\}$ that dominates G.

4. There is a deducible graph $G' \geq G$.

Proof: (1) This statement has already been proved (Lemma 1 and subsequent definitions).

(2) Let N_1 and N_2 be the (free) nodes of G_1 and G_2 respectively that are double-bar connected in G; let L_1 and L_2 be their respective associated literals. Since the resultant C_1' of G_1' subsumes the resultant C_1 of G_1, we have $C_1' \lambda \subset C_1$ for some λ. Moreover, every literal in C_1 is free in G except L_1, so the only way C_1' could fail to subsume C is if L_1 is a member of $C_1' \lambda$ (i.e., $L_1 = L_1' \lambda$ for some literal L_1' in C_1'). In that case G_1' can be double bar connected to G_2 on the nodes corresponding to L_1' and L_2, and the graph obtained clearly dominates G. (Should G_1' have more than one free node having L_1' as an associated literal we must first dot-connect those nodes in G_1'.)

(3) To simplify the discussion, we first replace the subgraph G_1 in G by an initial graph for its resultant C_1. We let $\hat{G}$ be the graph so obtained; clearly $\hat{G}$ has the same resultant C, and $\hat{G}$ and is constructed from initial clauses in $(S - S_1) \cup \{C_1\}$. Our proof is by induction of the total number of links in $\hat{G}$ (dotted links plus double-bar links). Since $\hat{G}$ is a graph, $\hat{G}$ is constructed in one of four ways:

(a) $\hat{G}$ is an initial clause which is necessarily C_1. In this case the number of links is 0 and we take G' = the graph of D.

(b) $\hat{G}$ is formed from a graph G_2 by adding a dotted link. By induction, since G_2 has fewer links than $\hat{G}$, there is a graph $G' \geq G_2$ containing initial clauses from $(S - S_1) \cup \{D\}$. Since $G_2 \geq \hat{G}$ ($\hat{G}$ is a factor of G_2) we have $G' \geq \hat{G}$.

(c) $\hat{G}$ is formed from two subgraphs G_2 and G_3 by adding a dotted link. (Note that G_2 and G_3 each dominate $\hat{G}$.) If G_2 is the subgraph containing C_1 (exactly one of G_2 or G_3 contains C_1), we take G' to be the graph satisfying the theorem for G_2. However, $G' \geq G_2 \geq \hat{G}$ so G' satisfied the theorem for $\hat{G}$.

(d) $\hat{G}$ is formed from G_2 and G_3 by adding a double bar link. Again, assume C_1 is contained in G_2, and let G_2' be the graph satisfying the theorem for G_2. By part (2) of Theorem 1, either $G_2' \geq \hat{G}$ in which case we set $G' = G_2'$ or else G_2' and G_3 can be double bar connected to produce a graph $G' \geq \hat{G}$. Since G_3 is constructed from $S - S_1$, and G_2' from $(S - S_1) \cup \{D\}$ we see that G' satisfies the theorem in either case.

(4) (We again do a proof by induction on the total number of links in G).

(a) If G is an initial clause it is deducible.

(b) If G is formed from G_1 by adding a dotted link, let G' be the deducible graph for G_1. But $G' \geq G_1 \geq G$ so G' is a deducible graph for G.

(c) If G is formed from G_1 and G_2 by adding a dotted link, let G' be the deducible graph for G_1. Again, $G' \geq G_1 \geq G$ so G' is a deducible graph for G.

(d) If G is formed from G_1 and G_2 by adding a double bar, then again by induction let G_1' and G_2' be the deducible graphs for G_1 and G_2 respectively. Now by Part (2) either $G_1' \geq G$ in which case we set $G' = G_1'$, or $G_2' \geq G$ in which case we set $G' = G_2'$, or G_1' and G_2' can be double bar connected to generate a graph $G' \geq G_1$. In this last case, we must dot-connect any nodes of G' containing identical associated literals in order that this last step constitutes a valid deduction.

Observations

1. Sometimes a deducible graph may be generated only by performing the resolutions in one order; attempting a different order causes literals to collapse (by the induced partition) and thus disappear from the graph. For example, consider the deductions possible from the three clauses C_1, C_2, and C_3, where,

$$C_1 = \{\sim P(x, a), \quad R(x)\}$$

$$C_2 = \{P(b, z) \ , \ Q(z, b)\}$$

$$C_3 = \{P(u, a) \ , \sim Q(a, u)\} \quad .$$

If we start by resolving C_1 with C_2, we can get (by Figure 6a) $\{P(b, a), R(b)\}$. If we instead start with C_2 and C_3, the deducible graph has an induced dotted link and we get simply P(b, a) (Figure 6b), or if we wish, R(b) (Figure 6c).

This phenomenon does not contradict the theorem, which merely asserts that the resultant clauses are the same whenever the graphs are the same, which they are not in Figure 6. Note that if we use the

pure graph operation of double-bar link, rather than resolution, we can construct Figure 6a in any order. Moreover, whenever we get collapses in deduced graphs we end up with a stronger clause, i.e. one that subsumes the clause obtained by resolving in a sequence that avoids the collapse.

Problems such as this order-dependence of deductions would have been avoided if we had defined a clause by a graph and an inference step by a double-bar link; such an approach would have resulted in a somewhat more elegant presentation. However, the idea that a clause is a set and resolution is an inferential operation upon sets is well entrenched in the literature (and in various computer implementations). Therefore, this paper has taken the more complicated approach of explicitly distinguishing deduced graphs and collapsed literals.

2. Any tautological subgraph may be eliminated from a non-tautological graph without weakening a deduction, i.e. there exists a graph without the tautological subgraph whose resultant subsumes the resultant of the original graph. (A tautological graph is one whose resultant contains a pair of complementary literals.) Since the entire graph is nontautological, there must exist a subgraph G_1 double-bar linked to the tautological subgraph $\mathfrak{J}$ on one of the troublesome literals; but then G_1 itself dominates the graph consisting of $\mathfrak{J}$ and G_1, and by the theorem may replace it in any larger graph. For example, consider the propositional deduction of Figure 7. This graph may be deduced with no problems by doing the lower resolution first. If

the upper resolution is done first, the tautological subgraph $\mathcal{J}$ with resultant $\{p, \sim p, s\}$ is deduced. However, note that the clause $D = \{p, r\}$ subsumes the resultant of the entire graph $\{p, r, s\}$.

V STRATEGIES AND REORDERING THEOREMS

In this section we prove the completeness of a new strategy for constructing resolution proofs. (Remember that we use "resolution" to mean optional factoring followed by simple resolution.) This strategy severely limits the alternative next steps available at each stage in a proof by superimposing several of the constraints described by other workers. Our approach is to show, constructively, how to transform any given graph into a dominating graph that can be generated by a proof satisfying the constraints.

The distinction between resolution with an initial clause, and resolution with a clause generated by previous resolution steps, is an important aspect of the class of theorem-proving strategies with which we are concerned. Consider a deducible graph containing several dotted lines. By Theorem 1, the order of operation that generated the graph is unimportant, and therefore any dotted lines between literals in the same initial clause can be produced by the factoring part of a resolution operation at the time that clause is introduced into the proof. On the other hand, a dotted line between literals in <u>different</u> initial clauses (a d-merge node) cannot be generated by a deduction until a subgraph has been constructed that contains all the parent clauses (because step 3 in the definition of a graph can never be used when the graph is generated by a deduction). This is the source of most of the complication in the following presentation, and the reason why d-merges are important in proof strategies.

Dot Reduction

The following lemma states that any particular dot-connected group in a graph may be split.

Lemma 2: Let G be the graph formed by a double-bar connection between a literal L in graph G_1 and the dot-connected node $\mathcal{L} = \{L_1, \ldots, L_n\}$ of group of circles in graph G_2. Then the graph G', obtained from G by removing L_1 from $\mathcal{L}$, double-bar connecting L in G_1 to L_1, and double-bar connecting the remainder of $\mathcal{L}$ to the copy of L in $\hat{G}_1$ (a new variant copy of G_1), exists and dominates G (see Figure 8a, c).

Proof: If we factor any clause together with an alphabetic variant of itself, we obtain essentially the same clause again. Therefore graph G^*, obtained by replacing subgraph G_1 in G by such a factored pair of variants, is equivalent to G (Figure 8b). Ignoring signs as usual, the relevant subset of $\mathcal{E}_{G^*}$ affected by the modification that is required by the lemma is $\{\{L_1, L_2, \ldots, L_n, L, \hat{L}\}, \{L', \hat{L}'\}\}$. The corresponding subset of $\mathcal{E}_{G'}$, is $\left\{\{L_2, \ldots, L_n, \hat{L}\}\{L_1, L\}\right\}$. Since the remainders of $\mathcal{E}_{G^*}$ and $\mathcal{E}_{G'}$ are identical, $\mathcal{E}_{G'} \leq \mathcal{E}_{G^*}$, therefore $G' \geq G^*$, and thus, $G' \geq G$.

Note: Since the ancestors of a d-merge literal in initial clauses form a dot-connected group in a graph, this lemma will be useful for eliminating or reducing the complexity of d-merges in deducible graphs.

Fishtail Deductions

The proof method of our main theorem, Theorem 2 below, will be induction upon the number of merges in a graph.

Our next lemma establishes a strong condition upon the structure of a proof that generated a graph containing no merges.

Definition: A fishtail construction of a graph G from a set S of initial clauses (including possible repetitions) is a finite sequence of graphs G_1, G_2, ... G_m such that

(1) G_1 is the graph of a clause from S, called the starting clause of the construction.

(2) G_i, $1 < i \leq m$, is the graph obtained from G_{i-1} and (the graph of) a member C of S by first dot-connecting a group of nodes in G_{i-1}, then dot-connecting a group of nodes in C and finally double-bar connecting the resulting nodes. (The dot-connecting could be a vacuous operation in either case.)

(3) G_m is G.

Definition: A fishtail deduction of a clause C from a set S of initial clauses is a finite sequence of clauses B_1, B_2, ..., B_n such that

(1) $B_1 \in S$ and is called the starting clause of the deduction

(2) B_i, $1 < i \leq n$, is the result of resolving B_{i-1} with some member of S. (Remember that "resolving" means simple resolution after optional factorings of both parents.)

(3) B_n is C.

Lemma 3:

(1) If there exists a fishtail construction of a graph G with clauses from S and starting clause B, whose resultant g-clause is C, then there exists a fishtail deduction of a clause C′ that generates a graph $G' \geq G$ with clauses from S and starting clause B.

(2) If G is a graph containing no merges (i.e., dotted links between literals in different initial clauses), then there exists a fishtail deduction that generates a graph $G' \geq G$ with any clause in G used as starting clause.

Proof:

(1) Starting with B, follow the steps of the fishtail construction. For each addition of a dotted link make a factoring step, and for each double-bar to a new clause make a simple resolution. By the argument in the proof of theorem 1 part 2, this is always possible unless the graph deduced at the previous deduction step already dominates the graph constructed at the current construction step--in which case no corresponding deduction step is necessary. This deduction sequence produces deducible graphs dominating the corresponding constructed graphs, so that at the final step the deduced graph G' will dominate the constructed graph G.

(2) Since there are no merges, each resolution link connects exactly two initial clauses and those two clauses are not connected in any other way, i.e. the graph contains no loops. Therefore, by theorem 1 part 1, a fishtail construction of G may be generated from any starting clause, and by (1) there exists a corresponding fishtail deduction of $G' \geq G$.

Note that the fishtail construction is a procedure for "growing" a graph G by starting with a single clause and "adding" additional initial clauses, thus forming successively larger subgraphs of G. Further growth of a subgraph in this way is impossible only when each connection from the subgraph to other parts of G is a double-bar link to a

merge--because each merge links at least two initial clauses, and a fishtail procedure requires adding initial clauses one at a time.

Examples: There clearly exists a fishtail construction of the merge-less graph of Figure 9a from any starting clause. In Figure 9b, any fishtail construction must start with either C_1 or C_2, since otherwise the dotted link cannot be introduced. Figure 9c is an example of a graph for which there does not exist a fishtail deduction. It represents a refutation, for example, of the four clauses $p \vee q$, $p \vee \sim q$, $\sim p \vee q$, and $\sim p \vee \sim q$. Either the left or right subgraph can be generated in fishtail fashion, but the growth process is then blocked by the double-bar link to a merge.

Main Theorem

Theorem 2: Let G be a graph generated by any deduction D of a clause C from an initial set S. Then there exists a graph G' and a deduction D' generating G' with the following properties:

(1) $G' \geq G$.

(2) The deduction D' contains a linear sequence of clauses B_1, B_2, ..., B_n such that

(a) $B_1 \in S$, and may be arbitrarily chosen to be any element of S whose graph occurs in G.

(b) For $1 < i \leq n$, B_i is a resolvent, one of whose parents (the "immediate" parent) is B_{i-1}. (The other parent of B_i is called the "far" parent.)

(3) Either the far parent of B_i, $1 < i \leq n$, is in S (the fishtail property), or the far parent satisfies all the following conditions:

(a) It is B_j for some j, $1 < j < i$.

(b) It is a d-merge, and the merge literal is the literal resolved upon.

(c) An instance of the resolvent B_i is identical to an instance of the clause obtained by deleting the literal resolved upon from the immediate parent B_{i-1}.

Proof: The strategy for the proof will be roughly this: We shall select successively larger subgraphs G_i of G. For each G_i, we shall show how to construct a deduction D_i' generating a graph $G_i' \geq G_i$ such that D_i' has properties (2) and (3) (assuming G is replaced by G_i in the statement of the theorem). Moreover, D_{i+1}' will be derived from D_i' simply by extending the deduction, i.e. by using the resultant C_i' of G_i' as a starting clause and "adding" (successively resolving with) initial clauses from G_{i+1} that did not occur in G_i. Eventually, for some j, G_j will be the complete graph G, at which time D_j' will be the required D'.

The proof is by induction on the number of merges in G. By Lemma 3 the theorem is true for any graph with no merges (because the fishtail property is satisfied). Now assume it is true for any graph with fewer than n merges, and assume G has n merges. Choose any starting clause B_1 in G. Let G_1 be the largest subgraph of G for which there exists a fishtail construction with starting clause B_1. C_1 is the resultant of G_1.

If $G_1 = G$, we are through, because by Lemma 3 there is an appropriate fishtail deduction. Otherwise, choose some free literal L (from a free node) in G_1 that is double-bar linked to the dot-connected node $\mathcal{L} = \{L_1, \ldots, L_n\}$, $n \geq 2$, of literals in G_1^*, a subgraph of G disjoint from G_1. Each L_i corresponds to a node from a different initial clause. (L must exist because the fishtail construction of G_1 cannot be continued.) Let G_5 be the subgraph of G consisting of G_1 and G_1^* (Figure 10a). We shall show how to construct a deduction D_5' generating a graph G_5' such that D_5' and G_5' satisfy the theorem (if G_5 is the entire initial graph G).

Reduce the merge $\mathcal{L}$ as described in Lemma 2, to produce $\tilde{G}_5 \geq G_5$ (Figure 10b). The required deduction D_5' generates a graph dominating $\tilde{G}_5$ and begins as follows: Construct the fishtail deduction starting from B_1 of a graph dominating G_1. Resolve the resultant C_1' with the initial clause in G_1^* containing L_1 (a fishtail step), forming G_2, whose resultant clause is C_2. Now consider G_3, the subgraph of $\tilde{G}_5$ shown in Figure 10c. It consists of G_2 and a subgraph of G_1^* (namely, G_1^* without the merge $\mathcal{L}$ or the initial clause containing L_1). Since the former subgraph G_2 is deducible (by the deduction of C_2 already described), we may think of it as if it were simply a single initial clause C_2.

Since G_3 now contains C_2 and a subgraph of G from which a merge $\mathcal{L}$ has been deleted, G_3 certainly has less than n merges. Therefore, by the inductive hypothesis there exists a deduction D_3' with starting clause C_2, generating a graph $G_3' \geq G_3$, that satisfies all the conditions (2) and (3) of the theorem.

We now ask, does G_3' dominate $\tilde{G}_5$? If so, the deduction thus far--the deduction of C_2 followed by D_3'--is the required D_5'. Unfortunately, G_3' does not necessarily dominate $\tilde{G}_5$. $\mathcal{E}_{G_3'} \leq \mathcal{E}_{G_3} \leq \mathcal{E}_{\tilde{G}_5}$, as required; and the literals associated with nodes of G_3' are a subset of those of $\tilde{G}_5$, except that G_3' may contain free literals of $\mathcal{L}$. Since $\tilde{G}_5$ has no free literals of $\mathcal{L}$, the resultant of G_3' does not necessarily subsume the resultant of $\tilde{G}_5$. We first factor together these troublesome literals, forming a single node. The rest of the proof is concerned with "getting rid of" this node.

One way to complete the deduction is to resolve C_3', the resultant of G_3', with a variant of the clause C_1 that appears earlier in the deduction, using the literals of $\mathcal{L}$ in C_3 against the variant $\hat{L}$ of L in C_1 (Figure 10b). However, this step would not generally satisfy condition (3b) of the theorem. Therefore we must be somewhat less direct.

Consider $\tilde{G}_5$. We may replace its subgraph G_3 by the single "initial" clause C_3'. (Note that the remainder of $\tilde{G}_5$ consists simply of $\hat{G}_1$.) Let G_4 be the largest subgraph remaining in $\tilde{G}_5$ for which there exists a fishtail construction with C_3' as starting clause. Let D_4' be the fishtail deduction of a graph dominating G_4(Lemma 3) (Figure 10d).

Finally, consider the case in which D_4' could not generate all of the subgraph $\hat{G}_1$. The fishtail process cannot continue only if the resultant of the deduction thus far must be resolved with a merged set of literals in the remaining part of the graph. Recall that D_4' actually includes deductions that start from B_1 and generate graphs dominating G_1, then G_2, G_3, and finally G_4. Let us number the steps of D_4'. Let C_4' be the name of the final clause deduced by D_4, and assume C_4' is deduced at

step k - 1. At step k, we would like to resolve C_4' against the resultant of a subgraph of $\hat{G}_1$ containing a particular merge (merge $\mathcal{M}$ in Figure 10b)--a nonfishtail step. However, since G_1 was generated by a fishtail deduction, that merge must have been formed in an immediate parent during that deduction; and that deduction D_1' is the first part of D_4'. Therefore a suitable clause for completing the deduction already exists as one of the $B_j, 1 < j < k$ (property (3a)). (In Figure 10e, we have drawn a <u>copy</u> $\hat{C}_k$ of this clause C_k to show the final resolution.) Note that the resolution must be performed upon a merge literal, thereby satisfying property (3b).

Let σ_r be the substitution required for this resolution at step k, and σ_m be a substituion that merges corresponding literals that are left in the two copies C_k and $\hat{C}_k$ appearing in the final resolvent $B_k = C_5'$. Then σ_m applied to B_k would make it identical to the clause obtained by applying $\sigma_r \circ \sigma_m$ to the immediate parent $B_{k-1} = C_4'$ after deleting the literal resolved upon, satisfying property (3c).

Finally, suppose G_5 is not the complete starting graph, and instead G contains additional subgraphs. Because of the definition of G_1, these additional subgraphs must contain merge literals double-bar linked to G_5. We may replace G_5 in G by the resultant C_5' of G_5', and continue the construction of the required deduction by "adding" to D_5', i.e. by treating C_5' as the starting clause.

<u>Tautologies</u>

For purposes of efficiency in a proof strategy based on Theorem 2, it would be desirable to add the following condition:

(2d) No B_i is a tautology.

Unfortunately, this is not true under the premises that G is generated by any deduction from any initial set S, and B_1 may be chosen arbitrarily. For example, if the middle clause $(p \vee q)$ is selected as B_1 in the propositional graph of Figure 11, then a tautological intermediate clause must be generated before obtaining the resultant $(\sim p \vee \sim q \vee r \vee s)$.

As we mentioned in the discussion after Theorem 1, the tautological subgraphs may be eliminated; in this particular example, either of the two terminal clauses subsumes the resultant. The problem here is that the designated starting clause, $p \vee q$, is essentially irrelevant to the desired result. If we are interested in refutations, i.e. deductions whose resultant C is the empty clause (graphs containing no free nodes), then following Anderson and Bledsoe (1970) we could require that the initial set S be minimally unsatisfiable. This means that S is unsatisfiable, but for any clause $C \in S$, $S - \{C\}$ is satisfiable. Therefore the graph of any refutation of S, including those whose tautological subgraphs have been eliminated, must contain an occurrence of every clause in S.

Thus we can assert the following theorem:

Theorem 2′: Let G be a resolution graph generated by a refutation from a minimally unsatisfiable set S. Then there exists a refutation generating G' that satisfies conditions (2) and (3) of Theorem 2, and also satisfies the conditions that no B_i is a tautology.

Proof: First eliminate all tautological subgraphs from G, producing a refutation graph $\hat{G}$. Since S is minimally unsatisfiable, any starting

clause chosen from S must occur in $\hat{G}$. Use $\hat{G}$ as the given graph in the proof of Theorem 2. Since that proof always replaces subgraphs by subsuming clauses and a tautology does not subsume any nontautological clause, no tautologies are introduced during the proof process.

VI CONCLUSIONS

Relation to Previous Work

Theorem 2 establishes that if a resolution deduction exists at all, then one exists that simultaneously satisfies several conditions. Clearly an assertion that a deduction exists satisfying only some of these conditions would be a corollary of Theorem 2. Therefore, we have just proven the completeness of all the following proof strategies:

1. Property (2a) establishes that any single initial clause occuring in a resolution proof is a sufficient set-of-support (Henschen, 1968).

2. Properties (2) and (3a) constitute the "ancestry filter" described by Luckham (1969).

3. Property (3c) is essentially the subsumption condition described by Loveland (1970)--and shown by Loveland to be compatible with ancestry filter and set-of-support.

4. Property (3b) is essentially a statement of Andrews' (1968) merge condition (shown by Andrews to be compatible with set-of-support).

Therefore the main results of this paper were to establish that all these strategies could be used simultaneously without losing completeness.

and to give some insight (by means of the resolution graph notation) into the significance of using each of the strategies.

Proof Strategy

Consider the following strategy $\mathcal{S}$ for proving a theorem from a set of axioms $\mathcal{A}$ by resolution:

1. Let S be the set of clauses obtained by placing all the members of $\mathcal{A}$, and the negation of the theorem, in quantifier-free conjunctive form.

2. Choose any clause in S that is known to be needed in the proof, usually a clause from the negation of the theorem, as the first clause B_1.

3. For each sequence $B_1, B_2, \ldots, B_i$, consider as <u>successor</u> clauses every B_{i+1} that satisfies all the properties (1) and (2) of Theorem 2 (and, if S is minimally unsatisfiable, the additional condition of Theorem 2′). This defines a tree of deductions.

4. Choose any algorithm for exhaustively searching the tree, e.g., breadth-first, or unit preference with level bound (Wos et al., 1964). Apply the algorithm.

5. A deduction of $\square$ (a "refutation") constitutes a proof of the theorem.

If any refutation containing the clause B_1 exists, one will be found by $\mathcal{S}$. Moreover, the nodes in the deduction tree have fewer successors than those of trees corresponding to less restrictive strategies; one hopes that this reduction in successors may reduce the total effort needed to find a proof. However, refutations that satisfy all the properties (1) and (2)

are usually longer than refutations that do not, and the relative sizes of the trees to be generated remain an open question.

The construction of a proof may be viewed as a tree-searching problem. Property (1), the linear format, essentially defines a class of deduction trees to be searched. The conditions of property (2) define the successor function. Theorem 2, and thus step 4 of S, have not treated the problem of selecting a good algorithm for attempting to find a refutation in the tree--and yet this algorithm may have a profound effect on the effectiveness of the search. Perhaps semantic heuristics, such as some kind of model partition strategy (Luckham, 1968) can be embodied into this algorithm without losing completeness. Another possibility is that suitable bounds can be found to enable practical use of the optimum tree-searching strategy A^* (Hart et al., 1968). Kowalski's paper (Kowalski, 1970) discusses this problem.

Further improvements in theorem-proving strategies might be obtained by studying the topological properties of resolution graphs. For example, the construction in the proof of Theorem 2 involves transforming a portion of a graph into one that is in a sense topologically simpler. In the extreme case of a graph with no loops, the stronger result of Lemma 3 is possible.

Theorem 2, property (2a) states that any clause that is used in a proof may be used as the top clause B_1, i.e., is a sufficient set of support. However, the choice of this clause may have a drastic effect on the length of the shortest deduction satisfying the rest of the

properties. For example, suppose $\mathcal{S}$ consists of the following clauses:

1. $\{P_x, Q_x\}$

2. $\{\sim Q_z, R_a, S_w\}$

3. $\{\sim R_x, S_a\}$

and the negation of the theorem of two clauses:

N1. $\{\sim P_a, Q_a\}$

N2. $\{\sim S_a\}$.

If N1 is chosen as B_1, we can get the refutation of Figure 12a, whose graph is Figure 12b. On the other hand, if $N_2 = B_1$, the refutation of Figure 13a, graphed in Figure 13b, is probably the shortest one that satisfies the conditions of Theorem 2. In general, given the resolution graph of a deduction, one may be able to establish on a purely topological basis the lengths of equivalent deductions that use particular strategies or support sets.

Another potential use for these graphs is as a basis to some new heuristic procedure for guiding the construction of a refutation. The graph of a refutation contains no free literals. The graph of an intermediate stage of a deduction therefore contains free literals that must be eliminated or "resolved away" in order to complete the proof. Since the graph structure contains more information than exists in its

resultant clause, perhaps a better strategy can be found than the usual unit preference or fewest component strategies.

Finally, let us consider the notion of "single connectedness." Wos et al (1967) define a resolution procedure to be singly connected provided no clash is generated in more than one way, where clash may be defined now as a clause whose resolution graph consists of n initial clauses, each resolved with a different literal from one initial clause of length greater than n. (By Theorem 1, all clashes that generated the same graph are equivalent.) A somewhat stronger property would be the following:

Definition: A resolution procedure is strongly singly connected iff it never produces two different deductions that generate the same resolution graph.

Since generation of the same (resultant) clause by alternate, equivalent deductions is a major cause of wasted effort in resolution procedures, strong single connectedness is an extremely desirable property. Perhaps the concept of resolution graphs can be the basis for a bookkeeping procedure for achieving this property in general. (We are aware of existing bookkeeping procedures that only work in unit resolutions.) Unfortunately, in order to test whether a proposed deduction is following a previously attempted path in this way, we would require an algorithm for testing whether the current resolution graph is a subgraph of any of a set of other previously established graphs. We know of no algorithm for this at present that is sufficiently efficient to be practical.

VII ACKNOWLEDGEMENTS

The research reported here was supported by the Advanced Research Projects Agency and the Rome Air Development Center under Contract F30602-69-C-0056, and is continuing under Contract NAS 12-2221 with the National Aeronautics and Space Administration and the Advanced Research Projects Agency. The authors wish to express their appreciation to C. C. Green, N. J. Nilsson, and M. I. Levin for many fruitful discussions that helped formalize the notions presented here.

REFERENCES

1. Anderson, R., and Bledsoe, W. W. A linear format for resolution with merging and a new technique for establishing completeness. J. ACM (in press).

2. Andrews, P. B. Resolution with merging. J. ACM 15, 3 (July 1968), 367-381.

3. Green, C. Application of theorem-proving to problem solving. Proc. Int'l Conf. on Artificial Intelligence (May 1969).

4. Green, C., and Raphael, B. The use of theorem-proving techniques in question-answering systems. Proc. 23rd Nat. Conf. ACM (1968), 169-181.

5. Guard, J., et al. Semi-automated mathematics. J. ACM 16, 1 (January 1969), 49-62.

6. Hart, P., et al. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. on Systems Sciences and Cybernetics, SSC-4, 2 (July 1968), 100-107.

7. Hayes, P., and Kowalski, R. Semantic trees in automatic theorem proving. Machine Intelligence 4, Edinburgh University Press, Edinburgh (1969), 87-101.

8. Henschen, L. J. Some new results in automated theorem proving. Report 261, Department of Computer Science, University of Illinois, Urbana (May 1968).

9. Kowalski, R. Search strategies for theorem proving. Machine Intelligence 5, Edinburgh University Press, Edinburgh (1970).

10. Loveland, D. W. A linear format for resolution. Proc. IRIA Sym., Lecture Notes in Mathematics, Springer-Verlag (in press).

11. Luckham, D. Some tree-paring strategies for theorem proving. Machine Intelligence 3, Edinburgh University Press, Edinburgh (1968), 95-112.

12. Luckham, D. Refinement theorems in resolution theory. Stanford University Artificial Intelligence Project Memo 81 (March 1969).

13. Raphael, B. Some results about proof by resolution. SIGART Newsletter, 14 (February 1969).

14. Robinson, J.A. A machine oriented logic based on the resolution principle. J. ACM 12, 1 (January 1965).

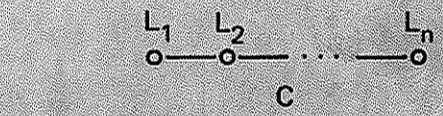
15. Robinson, J.A. A review of automatic theorem proving. Proc. Sym. in Applied Math., AMS, Providence, R.I. (1967).

16. Slagle, J. R. Automatic theorem proving with renamable and semantic resolution. J. ACM 14, 4 (October 1967), 687-697.

17. Wos, L., et al. The concept of demodulation in theorem proving. J. ACM 14, 4 (October 1967), 698-709.

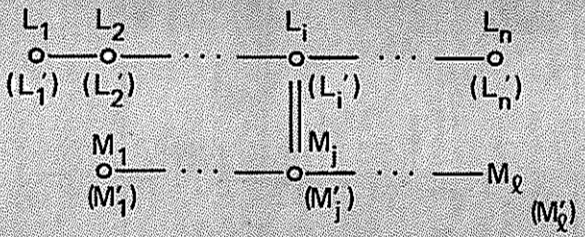
18. Wos, L., et al. The unit preference strategy in theorem proving. AFIPS Proc. FJCC 26 (1964), 616-621.

19. Wos, L., et al. Efficiency and completeness of the set of support strategy in theorem proving. J. ACM 12, 4 (October 1965), 536-541.



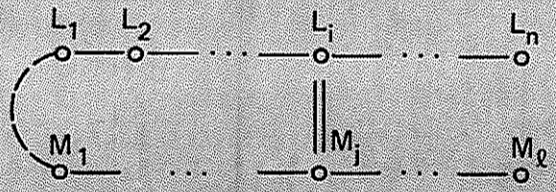
C

(a)



R(C, D)

(b)



R(C, D)

(c)

TA-710522-173R

FIGURE 1 ELEMENTARY RESOLUTION GRAPHS

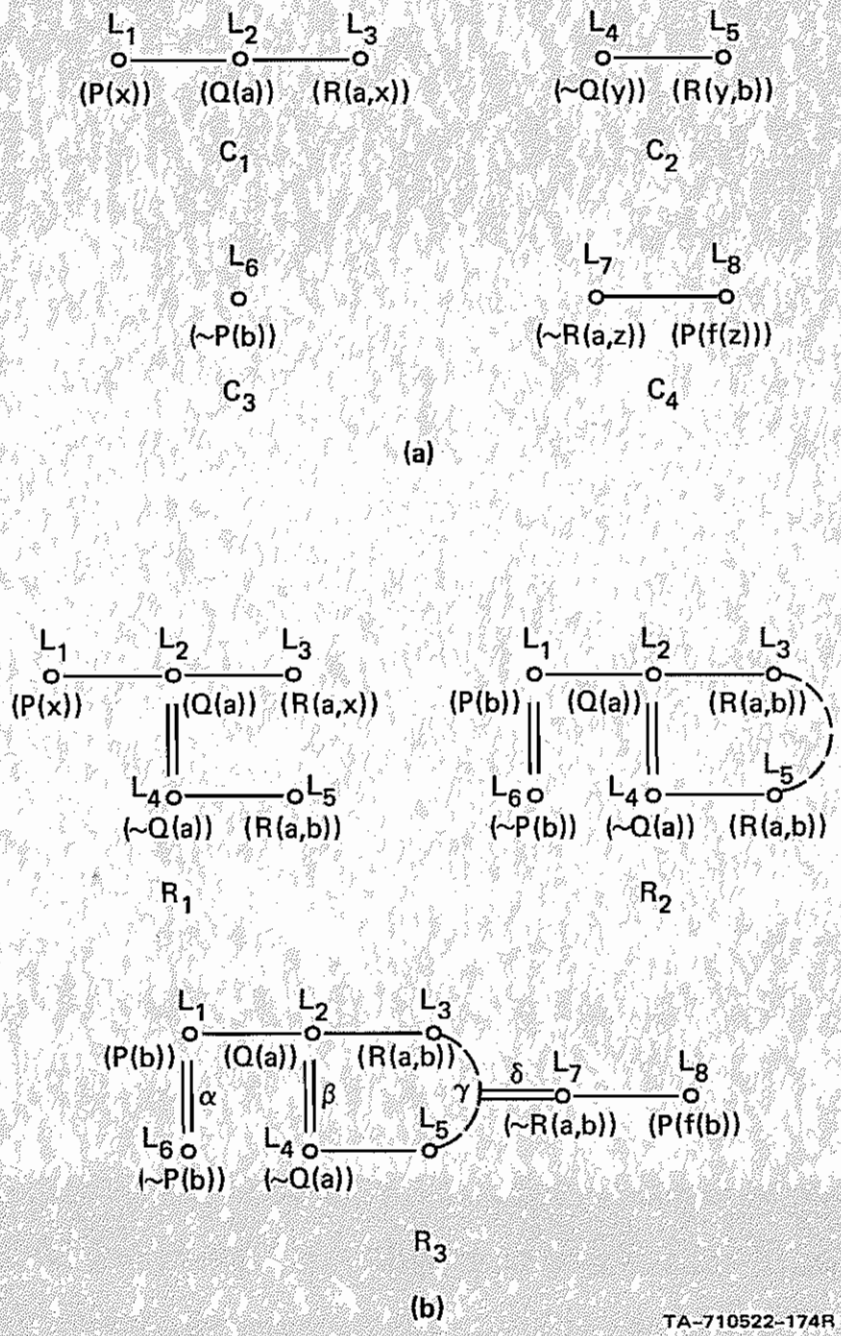
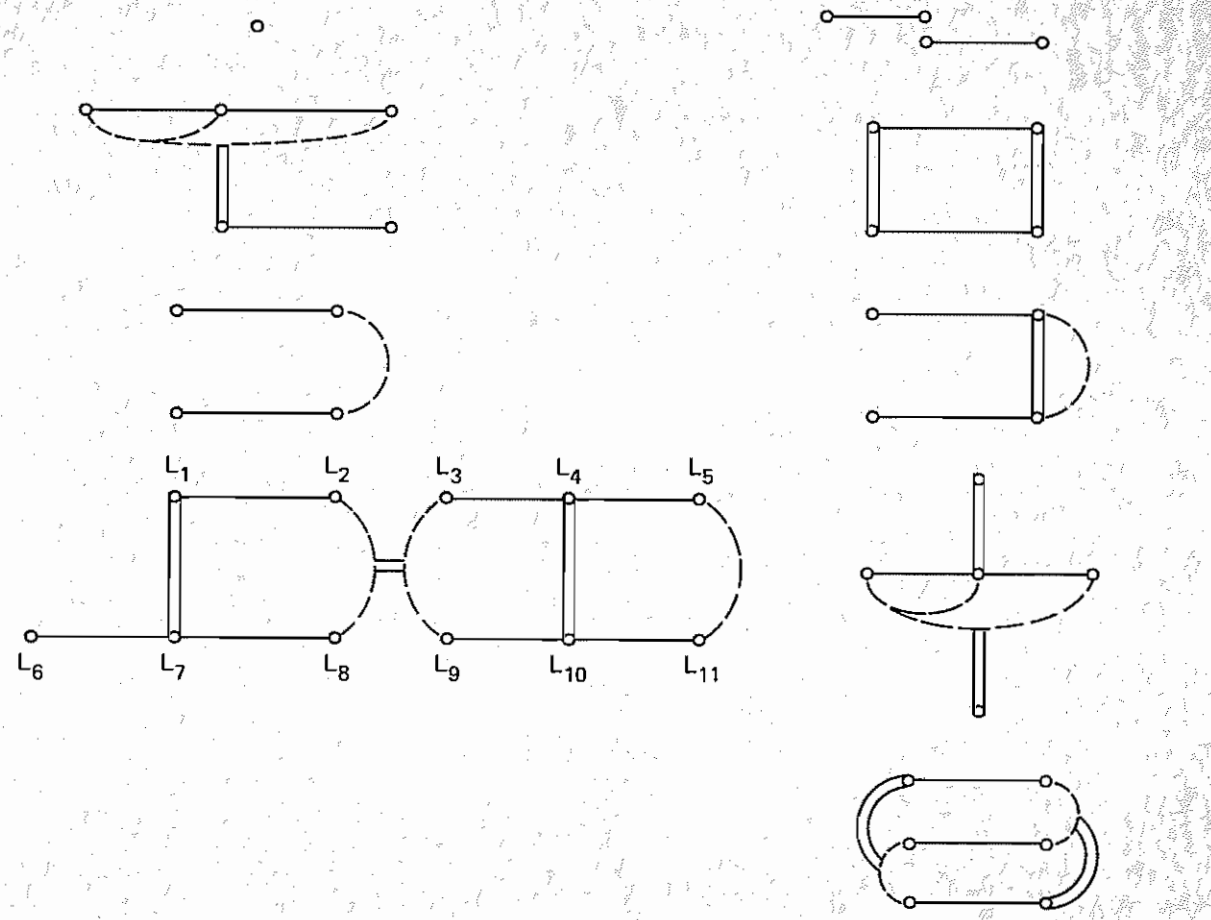


FIGURE 2 SUBSTITUTIONS DETERMINED BY GRAPH STRUCTURE



a. Possible Graphs

b. Structures That Cannot be Graphs

TA-710522-175

FIGURE 3 POSSIBLE AND IMPOSSIBLE GRAPHS

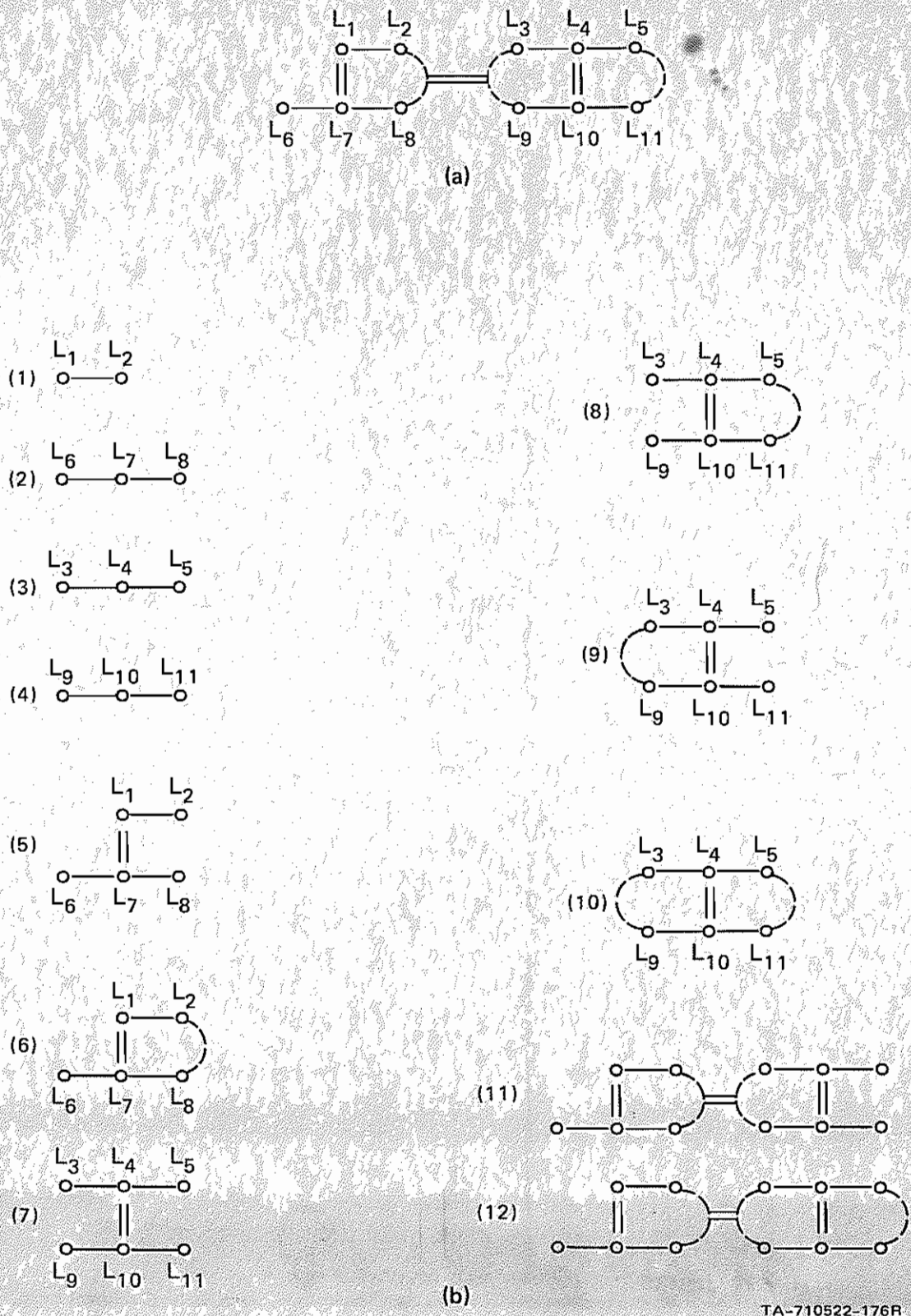


FIGURE 4 EXAMPLES OF SUBGRAPHS

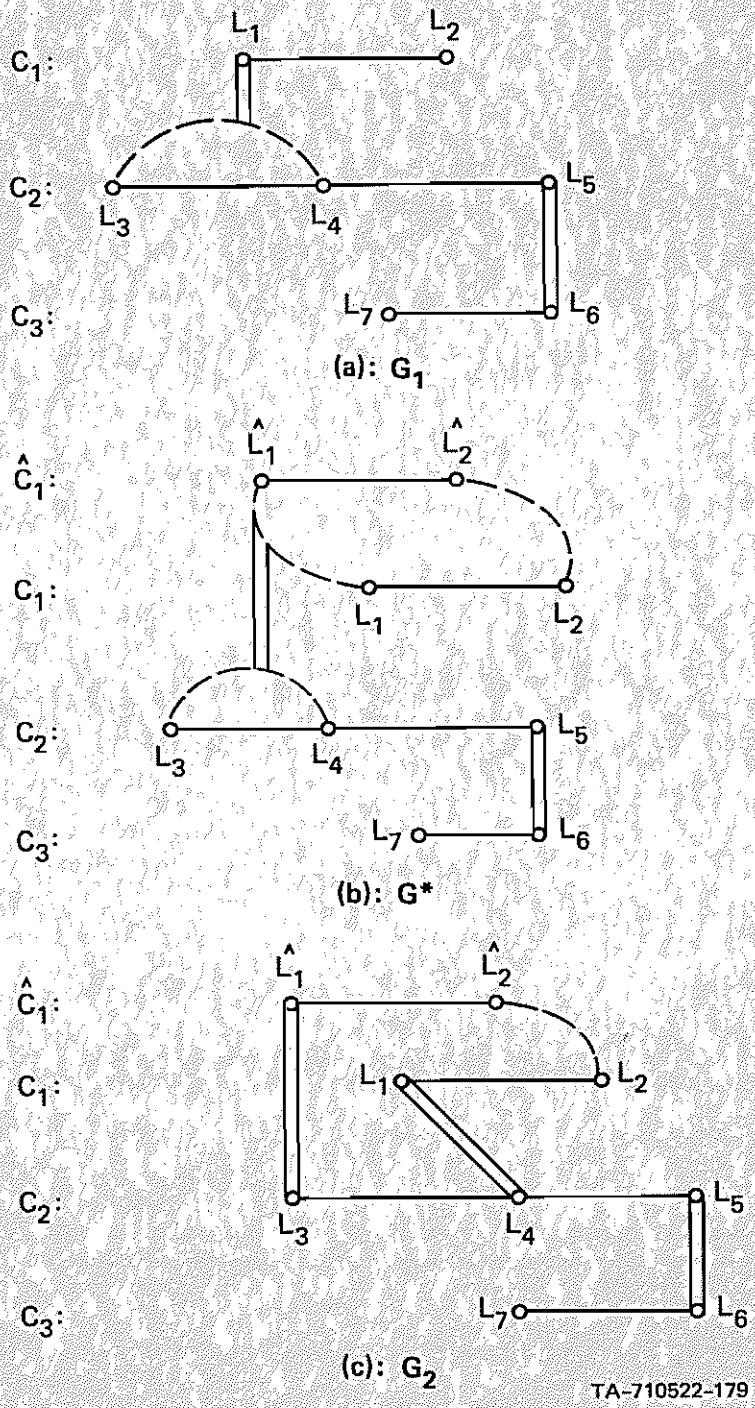


FIGURE 5 GRAPH DOMINANCE

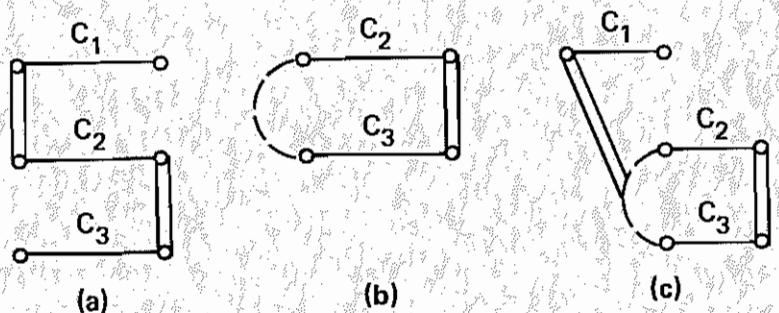


FIGURE 6 EXAMPLE OF ORDER DEPENDENCE

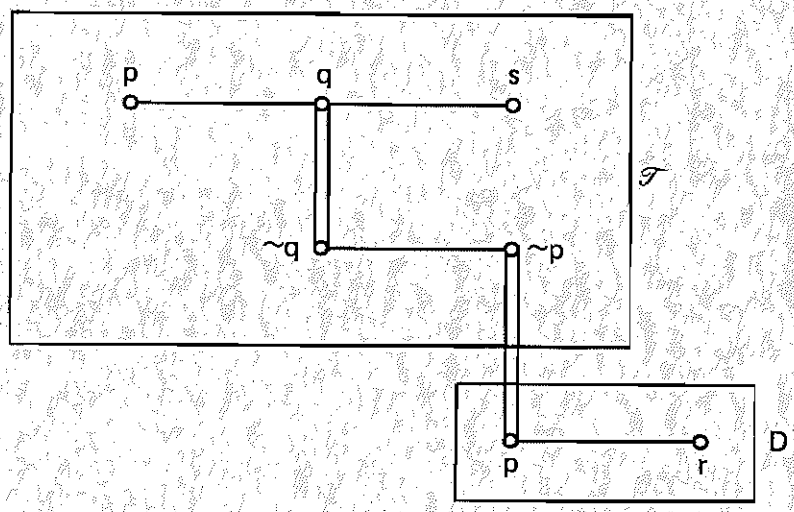


FIGURE 7 GRAPH WITH A TAUTOLOGICAL SUBGRAPH

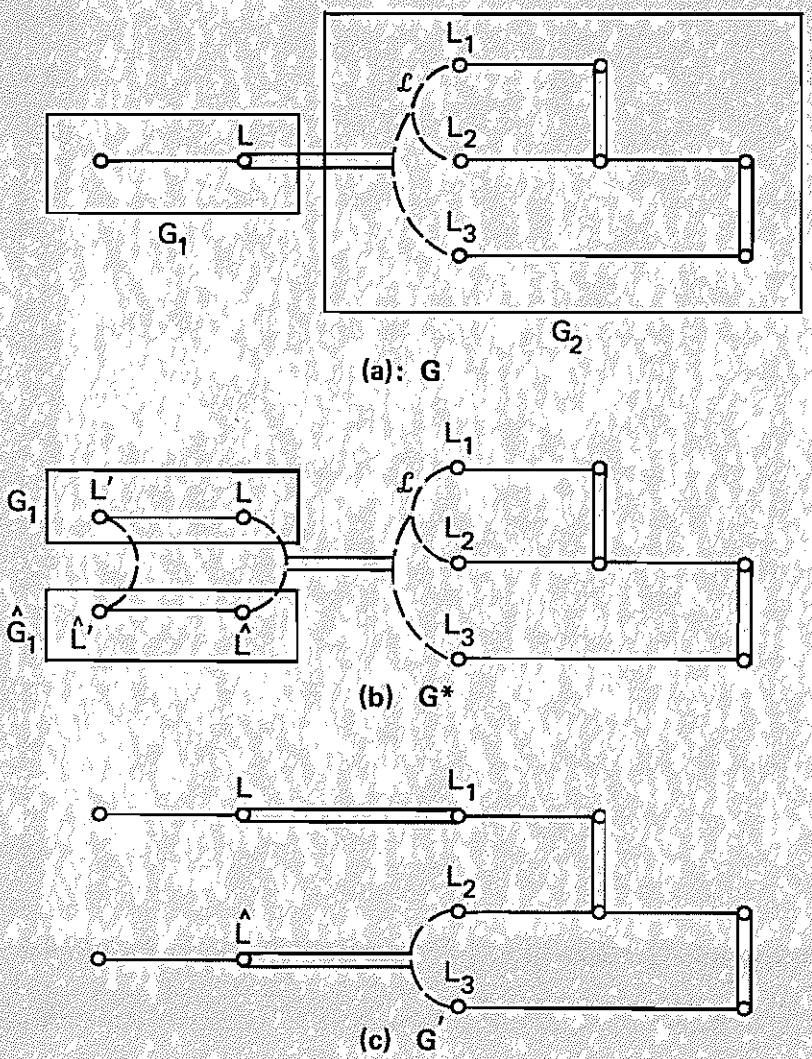


FIGURE 8 MERGE REDUCTION

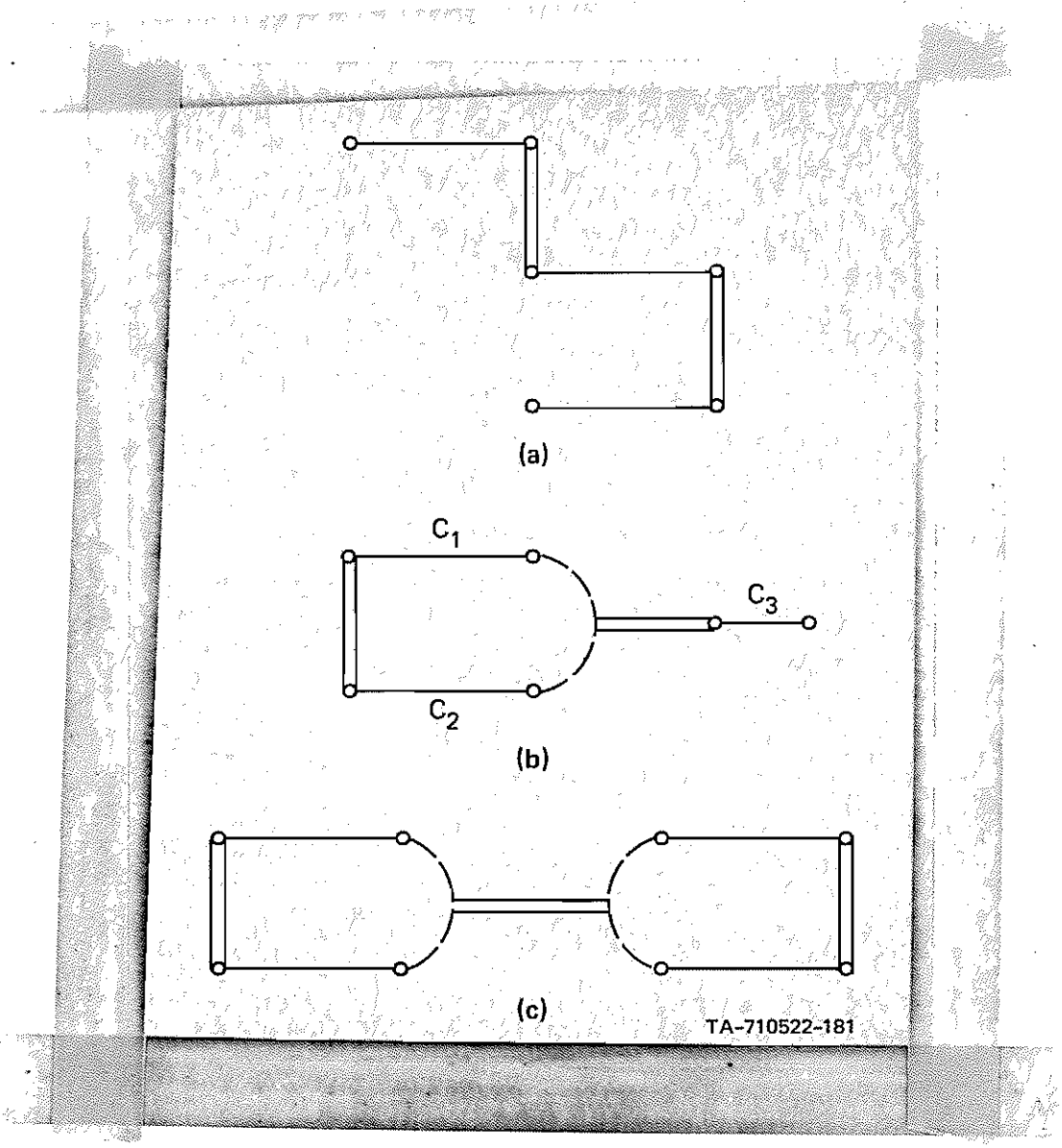


FIGURE 9 LIMITATIONS OF FISHTAIL DEDUCTIONS

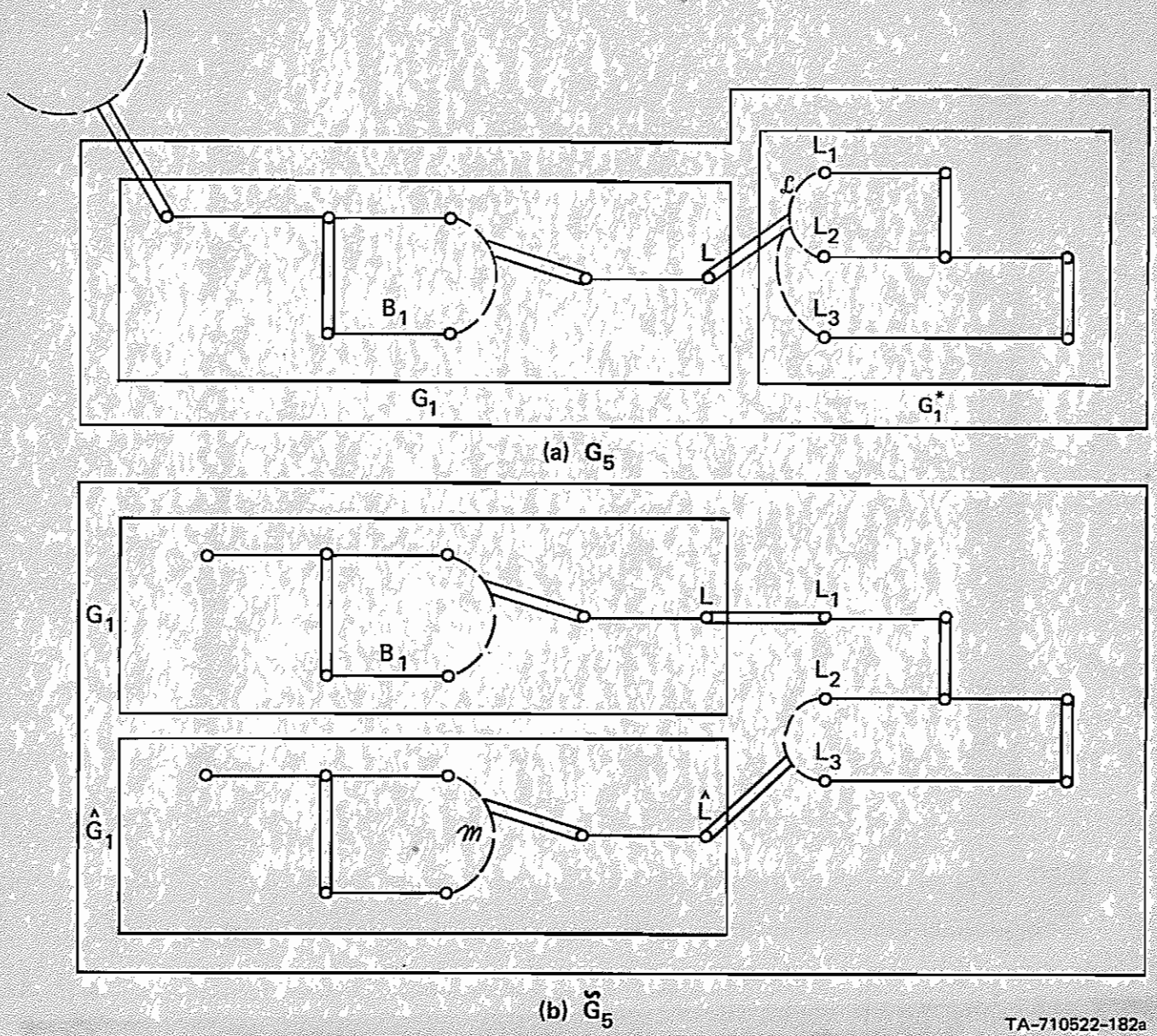


FIGURE 10 CONSTRUCTION OF A DEDUCTION SATISFYING THE CONSTRAINTS OF THEOREM 2

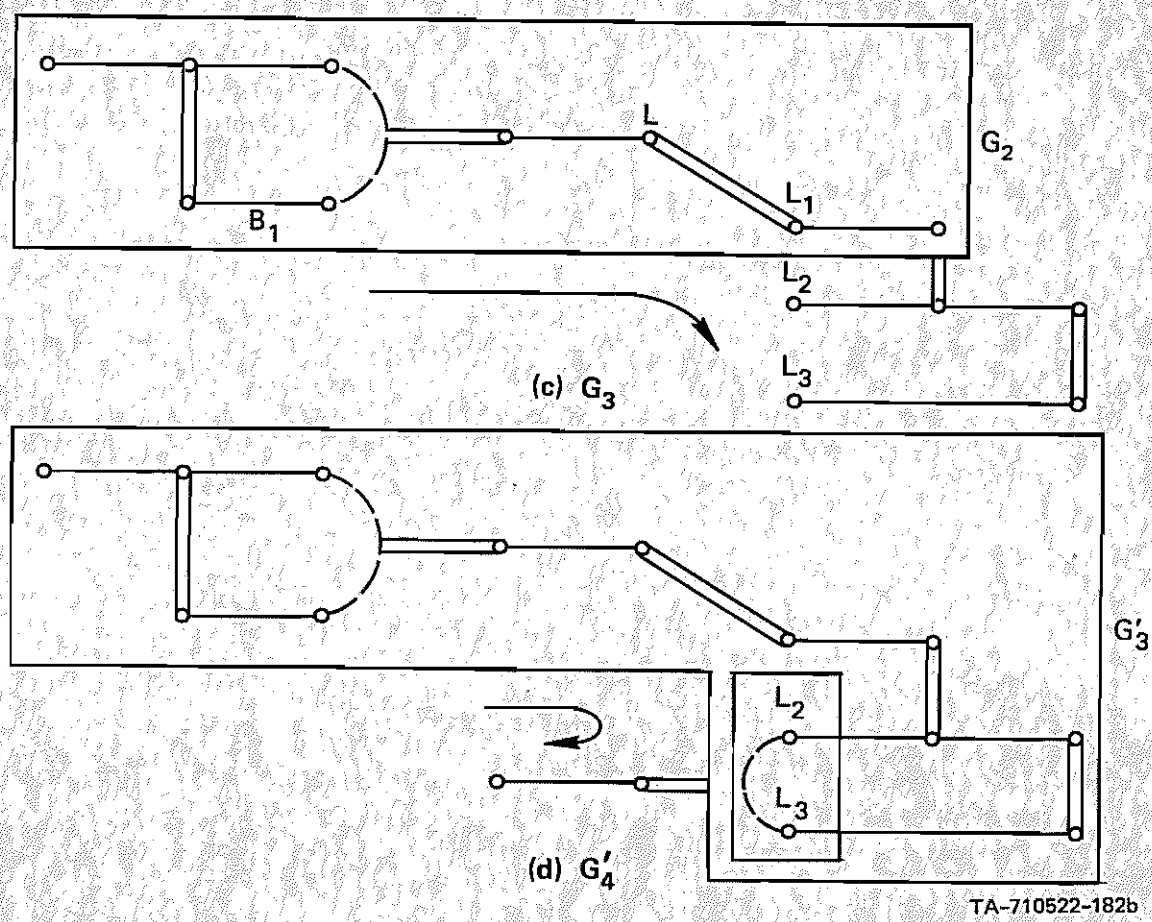
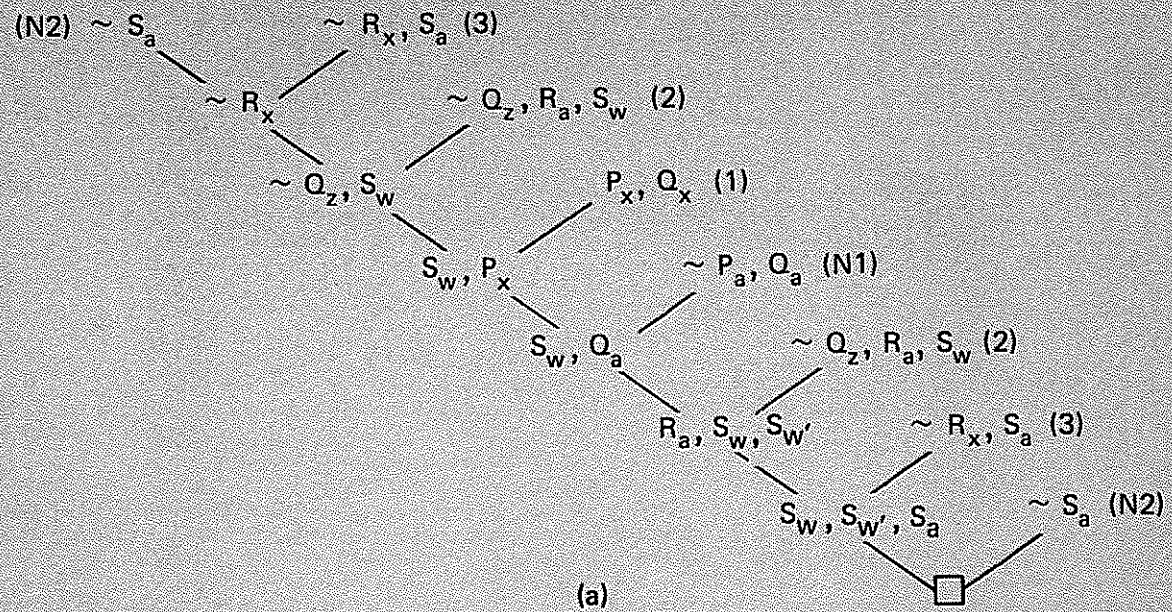
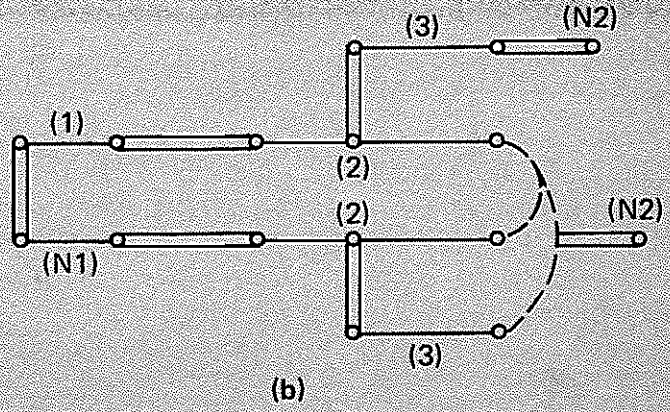


FIGURE 10 CONSTRUCTION OF A DEDUCTION SATISFYING THE CONSTRAINTS OF THEOREM 2 Continued





TA-710522-185

FIGURE 13 PROOF WITH POOR CHOICE OF STARTING CLAUSE

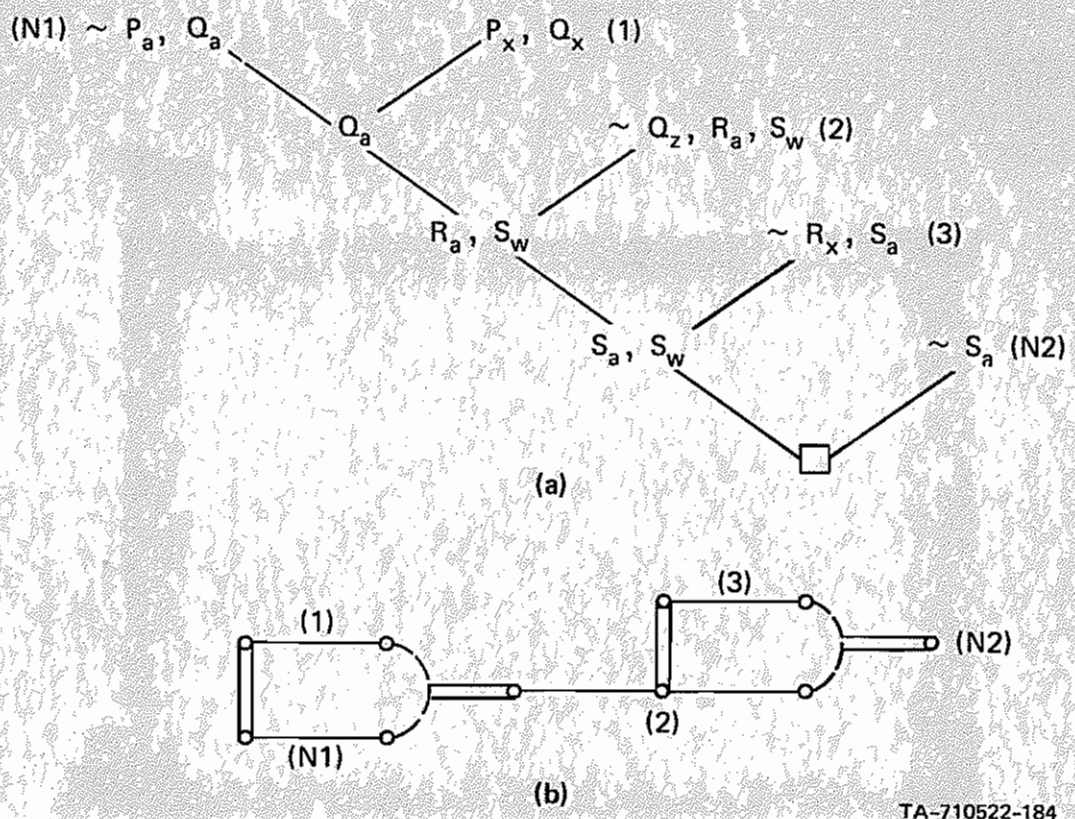


FIGURE 12 PROOF WITH GOOD CHOICE OF STARTING CLAUSE

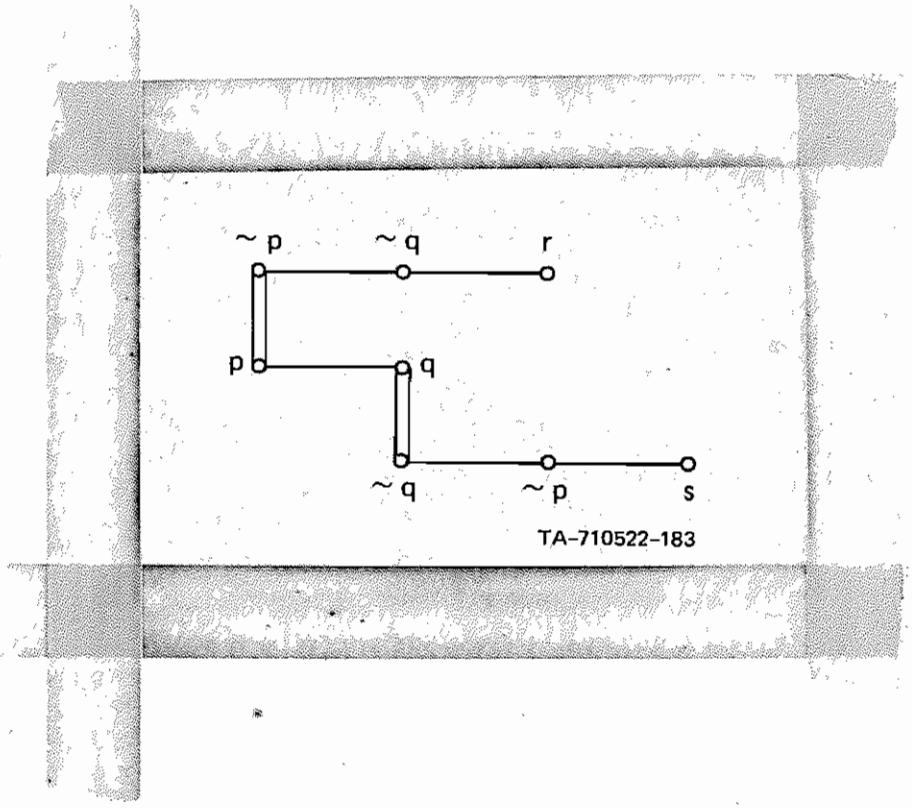
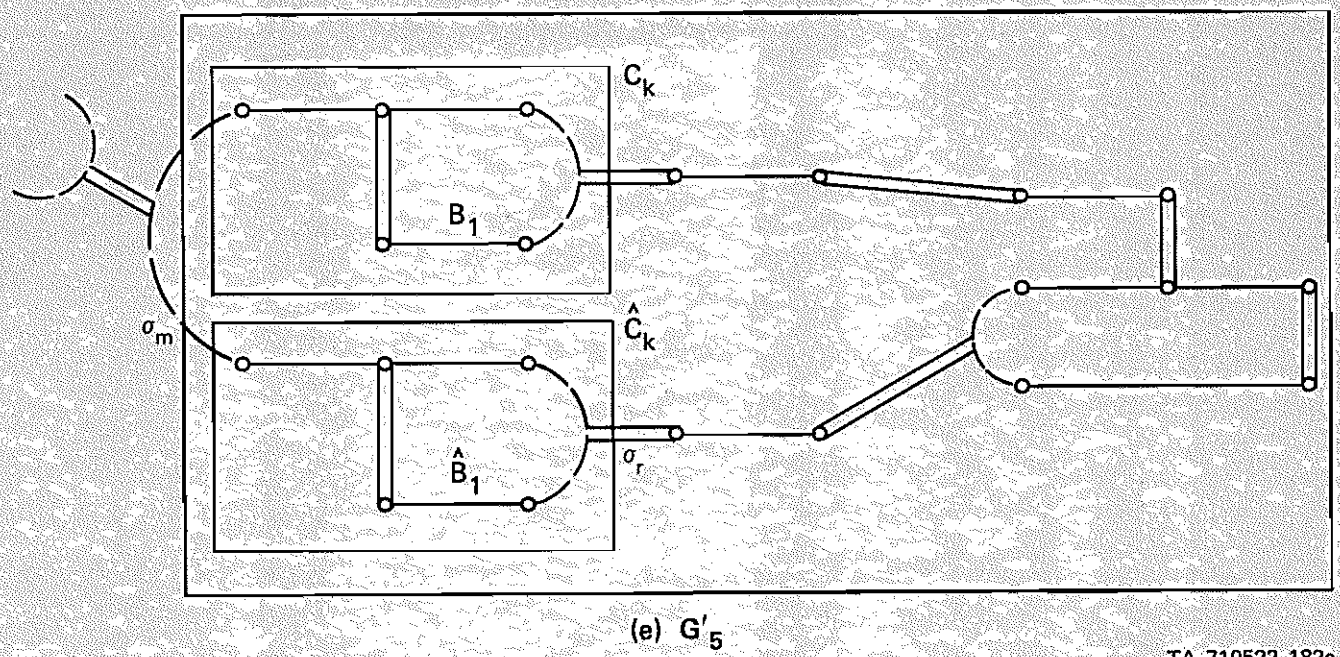


FIGURE 11 GRAPH WITH TWO TAUTOLOGICAL SUBGRAPHS



(e) G'_5

TA-710522-182c

FIGURE 10 CONSTRUCTION OF A DEDUCTION SATISFYING THE CONSTRAINTS OF THEOREM 2 Concluded