Resolution Strategies as Decision Procedures

WILLIAM H. JOYNER JR.

IBM Thomas J Watson Research Center, Yorktown Heights, New York

ABSTRACT. The resolution principle, an automatic inference technique, is studied as a possible decision procedure for certain classes of first-order formulas It is shown that most previous resolution strategies do not decide satisfiability even for "simple" solvable classes Two new resolution procedures are described and are shown to be complete (i e semidecision procedures) in the general case and, in addition, to be decision procedures for successively wider classes of first-order formulas These include many previously studied solvable classes The proofs that a complete resolution procedure will always halt (without producing the empty clause) when applied to satisfiable formulas in certain classes provide new, and in some cases more enlightening, demonstrations of the solvability of these classes A technique for constructing a model for a formula shown satisfiable in this way is also described

KEY WORDS AND PHRASES theorem proving, resolution, first-order logic, predicate calculus, decision procedure, solvable case, model, semantic tree

CR CATEGORIES 3 60, 5.21, 5 27

1. *Introduction*

Much of the previous research in automatic theorem proving (here taken to be the study of algorithms for demonstrating the unsatisfiability of first-order formulas) has centered on the resolution method of J.A. Robinson [28]. This technique operates on a set of clauses (each a set of signed atomic formulas) corresponding to a formula of first-order logic. The method is a semidecision procedure· Given an unsatisfiable formula, it is guaranteed to generate the empty clause, demonstrating the unsatisfiability. By well-known results of Church [5] and Turing [32], there can be no *general decision procedure*, which would demonstrate the satisfiability or unsatisfiability of any first-order formula submitted. There are, however, well-known and easily specified classes of formulas, called *solvable classes*, for which the question of satisfiability or unsatisfiability can be effectively decided; many solvable and unsolvable classes have been extensively investigated.

Since Robinson's result, most research in automatic theorem proving has been concerned with the development of strategies for resolution which maintain its completeness while attempting to hasten the production of the empty clause when given an unsatisfiable formula. Performance of these methods on satisfiable formulas is rarely considered (presumably because they cannot be guaranteed to halt when given such sets of clauses). But complete resolution procedures may terminate without generating the empty clause by reaching a point when no new clauses are

This paper is based on parts of the author's Ph D thesis at Harvard University, Cambridge, Massachusetts [11] A preliminary version was presented at the Fourteenth Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, October 15-17, 1973 [12]

This work was supported in part by the National Science Foundation under Grant GS-2301 and in part by the Advanced Research Projects Agency of the Department of Defense under contract F19628-71-C-0174.

Author's address Computer Sciences Department, IBM Thomas J Watson Research Center, P O Box 218, Yorktown Heights, NY 10598

generated; this implies the satisfiability of the input formula. This fact is used systematically below to develop new restrictions on resolution which assure its completeness in the general case and make it a decision procedure for a number of solvable classes

Unrestricted resolution (the original technique of Robinson) itself decides satisfiability for certain classes. For example, formulas which have as matrix a conjunction of signed atomic formulas produce sets of clauses in which each clause is a singleton From such a set, resolution either will produce the empty clause immediately or be able to generate no clauses, and so will always terminate on a formula of this kind However, it is shown below that resolution, even using previous strategies, is not a decision procedure for most solvable classes We show that two factors must be controlled to assure a resolution-type decision procedure: the increase of nesting of functional terms in resolvents, and the increase in size of clauses. Limitation of only one of these types of growth for a class of formulas and the completeness of a resolution procedure are not sufficient to assure the solvability of the class.

The resolution decision procedures developed below are based on a previous technique, the A-ordering strategy of Kowalski and Hayes [15], and are proved complete using their device of semantic trees. A particular A-ordering is defined which limits the growth of functional nesting in resolvents where possible To limit the growth of the number of literals in resolvents, a new technique called *condensing* is introduced; this consists of replacing each clause generated by the smallest instance of the clause which subsumes it.

Most previous decision procedures for solvable classes generate models for formulas which they demonstrate to be satisfiable. A method is given below for producing such a model (over the Herbrand universe) for a formula shown solvable by a resolution decision procedure, using a semantic tree and the finite set of clauses with which the procedure terminates. This is in a sense dual to previously developed methods of extracting proofs from resolution derivations of the empty clause [22]

2 *Review of Resolution Terminology*

The terms which describe the resolution technique have often been used by different authors in different ways. This section clarifies certain terminology used below, and introduces a small amount of new notation. Some familiarity with first-order logic and resolution deduction is assumed; for a more complete description of the method, see [4, 25, 28].

A first-order formula to be tested for unsatisfiability by resolution is first placed in prenex normal form, with its matrix in conjunctive normal form. Quantifiers are then deleted by replacing each existentially quantified variable by a term consisting of a new function symbol (a so-called *Skolem function*) whose arguments are the universally quantified variables preceding the existentially quantified variable in the prefix. The set of signed atomic formulas (called *literals*) of each conjunct of this functional form is called a *clause* Here we let $c(A)$, for any first-order formula A, be the set of clauses produced from A in this way.

Resolution operates on a set of clauses by forming new clauses, called *resolvents*, from pairs of clauses. Though production of resolvents is often decomposed into the two operations of resolving and factoring (see [25]), we retain here the original definition of Robinson [28] The two clauses are first *standardized apart* by renaming the variables in each with new, distinct variable names. If a set of atoms taken from positive literals in one clause and negative literals in the other can be *unified*, or made to be a singleton set by a substitution of terms for variables, then a resolvent may be formed The literals whose atoms are unified are the literals *resolved upon*, and the atom remaining in the singleton set after application of the most general such unifying substitution is the *resolved atom* The resolvent of the two clauses is formed by taking the union of all literals in the two clauses except those resolved upon, and applying this most general unifier.

Using this original definition of resolvent, we define a more general notion of a resolution proof procedure than that of Robinson. A function **R** from finite sets of clauses to finite sets of clauses is a *resolution procedure* if for a finite set of clauses S, $\mathbf{R}(S)$ is some finite subset of the union of S and the set (in general infinite) of instances of resolvents of pairs of clauses in S. This differs from the standard definition in that instances of resolvents (formed in some specified way) may be included in $\mathbf{R}(S)$, rather than only the resolvents themselves Also, only certain resolvents need be retained in $\mathbf{R}(S)$. These features allow the specification of various resolution strategies within this framework, such as those used as decision procedures below. Also allowed by this definition are procedures **R** such that some clauses in S are not in $\mathbf{R}(S)$ (such as strategies in which tautologies or subsumed clauses are deleted [25]). Strategies such that $S \subseteq \mathbf{R}(S)$ will be called *nondeleting*. Robinson's original resolution method, in which all possible resolvents are formed, is a resolution procedure under this definition and will be denoted $\mathbf{R}_1$.

If $\mathbf{R}^0(S)$ is defined as some (specific) finite set of instances of clauses in S (usually as the set S itself), and $\mathbf{R}^{n+1}(S)$ as $\mathbf{R}(\mathbf{R}^n(S))$, then the *application of* **R** *to* S is the construction of the finite sets of clauses $\mathbf{R}^i(S)$ for $i = 0, 1, \cdots$. The goal in this application is to produce the empty clause, written $\square$. Any set of clauses containing $\square$ is unsatisfiable, and if S is a satisfiable set of clauses then $\mathbf{R}(S)$ is satisfiable. (**R** is said to be *sound.*) Therefore if $\square \in \mathbf{R}^n(S)$ for some $n \geq 0$, S must be unsatisfiable. When this condition occurs, the application of **R** is said to *produce the empty clause* from S; the application then halts. If **R** produces the empty clause from any unsatisfiable set S, **R** is *complete* The completeness and soundness of a resolution procedure **R** are expressed by:

$$\square \in \mathbf{R}^n(c(A)) \text{ for some } n \geq 0 \leftrightarrow A \text{ is unsatisfiable.} \tag{2.1}$$

Though various techniques have been used to prove completeness for resolution strategies, the one most useful below is the method of semantic trees used by Kowalski and Hayes [15], which is discussed in Section 3.

The "normal operation" of a resolution procedure **R** to show a formula A unsatisfiable is to apply **R** to $c(A)$ until $\square$ is produced. (To show that a formula A is valid, this operation is applied to $\neg A$.) As pointed out by Robinson [28], however, a complete resolution procedure may halt without producing $\square$ by generating no new clauses at some resolution level, i.e if $\mathbf{R}^n(S) = R^{n+1}(S)$ for some n. If we assume that clauses that are variants (which differ only in the names of their variables) are not treated differently by **R**, then this halting condition can be weakened somewhat. Let us write $S \approx T$ for sets of clauses S and T if every clause in S has a variant in T, and conversely. Then an application of resolution procedure **R** is said to *halt without producing the empty clause* on S if $\mathbf{R}^n(S) \approx \mathbf{R}^{n+1}(S)$ and $\square \notin \mathbf{R}^n(S)$ for some $n \geq 0$. If this condition occurs, then $\square$ clearly can never be produced, for

$$\mathbf{R}^{n+2}(S) = \mathbf{R}(\mathbf{R}^{n+1}(S)) \approx \mathbf{R}(\mathbf{R}^n(S)) = \mathbf{R}^{n+1}(S) \approx \mathbf{R}^n(S),$$

and in general $\mathbf{R}^{n+i}(S) \approx \mathbf{R}^n(S)$ for $i = 1, 2, \cdots$ Therefore if **R** is complete, S must be satisfiable:

$$[\text{For some } n \geq 0,\ (\mathbf{R}^{n+1}(c(A)) \approx \mathbf{R}^n(c(A)) \text{ and } \square \notin \mathbf{R}^n(c(A)))] \rightarrow A \text{ is satisfiable} \tag{2.2}$$

In what follows, we will exploit this manner of halting in a systematic way

3. *Semantic Trees*

The method of semantic trees used by Kowalski and Hayes [15], who attribute it to Robinson [30], is used below not only to prove the completeness of resolution decision procedures, but also in Section 10 to extract models of formulas shown satisfiable. Therefore we briefly describe the method here. Nilsson [25] provides a fuller explanation

Let S be a finite set of clauses. The *Herbrand universe* of S is the set (in general infinite) of all *ground* terms (those with no variables) which can be constructed from the set of function symbols occurring in S, augmented by the constant a if no constants appear The *Herbrand base* of S is the set of atoms constructed from the predicate symbols in S and the terms of the Herbrand universe. Let $A_1, A_2, \cdots$ be some nonrepetitive enumeration of the Herbrand base of S. The (binary) *semantic tree* T corresponding to this enumeration is a binary tree (in general infinite) with elements of the enumeration and their negations labeling the edges. The two edges leaving the root are labeled A_1 and $\neg A_1$, and if either A_i or $\neg A_i$ labels the edge entering a node, then A_{i+1} and $\neg A_{i+1}$ label the two edges leaving that node With each node of T is associated the *assignment at* that node, which is the set of literals labeling the edges from the root to the node The assignment at the root is the empty set.

A clause C in S *fails* at a node of T if the complement of every literal in some ground instance $C\sigma$ of C is in the assignment at that node. A node is a *failure node* for S if some clause of S fails there but no clause of S fails higher in the tree (i.e. closer to the root). T is *closed* for S if there is a failure node for S on every branch of T. Note that a closed tree can have only finitely many failure nodes, and (unless the root is a failure node) must have at least one node, called an *inference node*, both of whose sons are failure nodes.

The method of proof of the completeness of a resolution procedure (such as $\mathbf{R}_1$) using semantic trees rests on two facts:

1. If S is unsatisfiable, every semantic tree T for S is closed for S. (Otherwise, a path with no failure nodes would determine a model for S)

2. The two clauses which fail immediately below an inference node form a resolvent which fails at the inference node. (This is because the complementary literals A_i and $\neg A_i$ labeling the branches leaving an inference node must appear in the instances of the clauses failing below the node.)

To prove $\mathbf{R}_1$ complete, we first note that since $S \subseteq \mathbf{R}_1(S)$, T can have no fewer failure nodes for S than for $\mathbf{R}_1(S)$. But at least two failure nodes for S (those below an inference node) are not failure nodes for $\mathbf{R}_1(S)$; the inference node itself, or some higher node, has replaced them So T has fewer failure nodes for $\mathbf{R}_1(S)$ than for S. Repeated applications of $\mathbf{R}_1$ continue to reduce the number of failure nodes, with T always remaining closed. Eventually T must have one failure node for some $\mathbf{R}_1^n(S)$, which must be the root since T remains closed. But $\Box$ is the only clause failing at the root, so $\Box \in \mathbf{R}_1^n(S)$.

In proving $\mathbf{R}_1$ to be complete in this way, any enumeration can be used to form the tree, since $\mathbf{R}_1$ allows formation of all possible resolvents. Below we show the completeness of more restrictive resolution strategies by a more judicious choice of this enumeration. We will say that a resolution procedure $\mathbf{R}$ is *complete via semantic trees* if $\mathbf{R}^n(S)$ is unsatisfiable whenever S is, and, in addition, if for every set of clauses there is a semantic tree T (corresponding to some enumeration of its Herbrand base) such that if C and D are any two clauses which fail immediately below a node of T, then some clause in $\mathbf{R}(\{C, D\})$ fails at that node. Kowalski and Hayes in [15] essentially show that a procedure complete via semantic trees is indeed a complete resolution procedure.

4. *The Decision Problem*

By a well-known result of Church [5] and Turing [32], there can be no *general decision procedure* for first-order logic; i.e. there is no algorithm which will determine whether any given formula is satisfiable or unsatisfiable. Thus for no complete resolution procedure **R** can (2 2) above be an implication in both directions Because (2 1) holds, a complete resolution procedure is said to be a *semidecision procedure* for first-order logic; i.e. it will always halt and produce □ when applied to an unsatisfiable set of clauses, but may continue forever if given a satisfiable set.

Because the halting condition in (2.2) cannot be guaranteed for every satisfiable A, most previous treatments of resolution strategies consider their performance on unsatisfiable formulas only. The goal of these strategies in most cases is to reduce the number of resolvents which must be generated from an unsatisfiable set of clauses before □ is produced. The halting condition (2.2), if tested for at all, occurs only accidentally. Below we investigate resolution strategies which are guaranteed to halt on sets of clauses with certain properties by using (2.2) in a systematic way.

The *decision problem* (for satisfiability) for an effectively specified class C of first-order formulas is the problem of finding an algorithm, called a *decision procedure*, which determines for each formula F in C whether or not F is satisfiable. C is *solvable* if it has a decision procedure, and *unsolvable* if not. The result of Church and Turing mentioned above is thus that the class of all first-order formulas is unsolvable.

Many solvable classes of first-order formulas, usually specified by simple syntactic features, have been studied in detail by logicians [2, 8, 9]. A number of them which will be considered in the following sections are listed in Table I. These are all subsets of the *pure* first-order predicate calculus, containing only formulas without function symbols and without equality. They are specified by the truth functional structure of the formulas, the form of the prefix in prenex normal form, the number of distinct arguments allowed for predicate letters, or a combination of these features. Here $\forall^*$ or $\exists^*$ is used to denote a string of any number (possibly zero) of universal or existential quantifiers, respectively. Much of the terminology used in the description of these classes is taken from the work of Dreben and Goldfarb [8].

Several of the classes in Table I are maximal solvable classes in some sense. Lowenheim, who showed the solvability of the monadic class in 1915, also showed at the same time that the addition of a single dyadic predicate symbol produced a *reduction class* (one whose solvability would yield a general decision procedure) [20]. The extension of the Herbrand class [10] to a matrix having a conjunction of binary disjunctions of literals (a so-called *Krom matrix*) was shown unsolvable by Orevkov in 1969 [26] and independently by Krom in 1970 [17]. The three prefix classes 3, 4, and 5 in the table are the only solvable classes specified by prefix alone. Their only possible prefix extensions are reduction classes: $\forall\forall\forall\exists$ (shown by Surányi in 1943 [31]) and $\forall\exists\forall$ (shown by Kahr, Moore, and Wang in 1962 [13]). The solvable extended Skolem class [7] has several solvable extensions; the simplest of these is the class with the same prefix of 6 in the table, such that for some fixed $n_0 \leq n$, (iii) becomes:

(iii′) all of $y_1, \cdots, y_{n_0}$ but none of $y_{n_0+1}, \cdots, y_n$.

Note that the definition of the extended Skolem class includes the Ackermann and Godel classes. Also, the Gödel class in a sense contains the Ackermann, since the

TABLE I Some Solvable Classes

1 Monadic	Only one place predicate letters appear
2. Herbrand	Matrix a conjunction of literals
3 Bernays-Schonfinkel	Prefix $\exists^*\forall^*$
4. Ackermann	Prefix $\exists^*\forall\exists^*$
5 Godel	Prefix $\exists^*\forall\forall\exists^*$
6 Extended Skolem	Prefix $\exists z_1 \quad \exists z_p \forall y_1 \quad \forall y_n \exists x_1 \cdot \quad \exists x_m$, and such that each atomic component of the matrix has among its arguments either (i) at least one of the x_i, or (ii) at most one of the y_i, or (iii) all of $y_1, \quad , y_n$
7 Maslov	Prefix $\exists^*\forall^*\exists^*$ and matrix a conjunction of binary disjunctions of literals

addition of a vacuous universal quantifier to an Ackermann formula produces an equivalent in the Godel class.

The classes of Table I are by no means all of the solvable classes which have been studied More complicated combinations of syntactic features produce more and more classes, some of which have been shown solvable. The literature on unsolvable classes and reductions of the decision problem is even greater.

5 *Resolution Decision Procedures*

Several authors have developed methods for systematically treating many of the solvable classes mentioned above; among these are the method of decision tables [9], the amenability method [8], and the inverse method [24]. The inverse method is claimed by Maslov to provide a decision procedure for a wide range of solvable classes, though the descriptions of the method and of its use as a decision procedure are obscure. In this section a way in which the resolution method can be refined to make it a decision procedure for certain classes is described.

Although resolution is one of the most widely studied complete refutation procedures for first-order logic, and many strategies aimed at improving its efficiency have been developed, little of this investigation has centered on the possibilities of resolution as a decision procedure. Kallick [14] did develop a resolution procedure, based on Friedman's algorithm [9], which was a decision procedure for the prefix class $\forall\forall\exists$, a subset of the Godel class of Table I. However, this algorithm was specialized to accept only formulas of this form (i.e. it was not a semidecision procedure for the general case), and it was not immediately extendable to other solvable classes. Our aim below is to produce a resolution strategy which is complete on the general case, is a decision procedure for a number of solvable classes, and provides some insight into the reasons for the solvability of those classes.

Recalling (2.1) and (2.2) above, we define a *resolution decision procedure* for an effectively specified class C of first-order formulas to be a complete resolution procedure **R** such that

$$[F \in \mathrm{C} \;\&\; F \text{ satisfiable}] \rightarrow [\mathbf{R}^{p}(c(F)) \approx \mathbf{R}^{p+1}(c(F)) \text{ for some } p \geq 0]. \tag{5.1}$$

Note that a resolution decision procedure for a class C will always halt when applied to a formula in C, either by generating $\Box$ (from an unsatisfiable formula) or by failing to generate any (essentially) new clauses at some level p (for a satisfiable formula). Thus a resolution decision procedure is a decision procedure as defined in Section 4. But since a resolution decision procedure is required to be complete, such a procedure has the stronger property of being (in addition) a semidecision procedure for the class of all formulas Also note that, unlike some previous decision methods, a resolution decision procedure **R** need not be given the information that a formula F submitted to it is in C to assure that it will halt when applied to $c(F)$; **R** will simply compute $\mathbf{R}^0(c(F))$, $\mathbf{R}^1(c(F))$, $\cdots$ until one of the two halting conditions occurs.

It is natural now to ask whether the unrestricted resolution method of Robinson, and previous resolution strategies, are decision procedures for any of the classes of Table I. It was pointed out in Section 1 that any resolution procedure will always halt when applied to a formula of the Herbrand class, which generates a set of unit, or one-literal, clauses. Most other known solvable classes (including all of the other classes listed above), however, are characterized by criteria other than their truth-functional structure. For these classes, unrestricted resolution is not a decision procedure.

As an example for some of the classes of the table, consider the following satisfiable formula:

$$F = \forall x \exists y((P(x) \vee P(y)) \;\&\; (\neg P(x) \vee \neg P(y))) \tag{5.2}$$

This formula is a member of the monadic class, the Ackermann class, and the Maslov class, and, by extension, of the Godel class and the extended Skolem class. Now $c(F)$

$= \{\{P(x), P(f(x))\}, \{\neg P(x), \neg P(f(x))\}\}$. If $\mathbf{R}_1$ is applied to $c(F)$, the clauses $\{P(x), \neg P(f(f(x)))\}$, $\{P(x), P(f(f(f(f(x)))))\}$, $\cdots$, $\{P(x), P(f\cdots f(x)\cdots)\}$, $\cdots$ are resolvents produced at successive levels of resolution. Since none of these clauses is a variant of a clause previously generated, $\mathbf{R}_1$ will not halt when applied to $c(F)$, and thus it is not a decision procedure for the solvable classes of which it is a member. Such previous resolution strategies as P_1-resolution [29], resolution with merging [3], linear resolution [19, 21], and S-L resolution (with any selection function) [16] also generate these clauses, and hence they are not decision procedures for these simple solvable classes.

Now consider the satisfiable formula

$$E = \forall x_1 \forall x_2 \forall x_3((P(x_1, x_2) \vee Q(x_2, x_3) \vee R(x_1, x_3)) \;\& \\ (\neg R(x_1, x_2) \vee Q(x_2, x_3) \vee \neg P(x_1, x_3))) \quad (5.3)$$

of the Bernays-Schonfinkel class. If $\mathbf{R}_1$ is applied to

$$c(E) = \{\{P(x_1, x_2), Q(x_2, x_3), R(x_1, x_3)\}, \{\neg R(y_1, y_2), Q(y_2, y_3), \neg P(y_1, y_3)\}\},$$

clauses of the form

$$\{P(x_1, x_2), Q(x_2, x_3), \cdots, Q(x_{n-1}, x_n), R(x_1, x_n)\}, \quad n = 5, 7, 9, \cdots,$$

are generated at successive levels. Thus $\mathbf{R}_1$ and, with some modification of the formula, several of the strategies mentioned above are not decision procedures for the Bernays-Schönfinkel class.

These examples illustrate the two factors which prevent resolution strategies from being decision procedures for solvable classes. Application of resolution to (5.2) shows *growth in nesting*, in which terms of increasingly greater functional height appear in the generated clauses. Note that in this example, clause size, in terms of number of literals, remains bounded by two at all levels. The clauses generated from (5.3) illustrate *growth in size*: Clauses with more and more literals are produced (although, in this example, no growth in nesting occurs).

Since formulas (5.2) and (5.3) are in several solvable classes, these examples indicate that limitation of only one of the types of growth by a complete resolution procedure for a certain class is not sufficient to make that method a resolution decision procedure for that class. However, all clauses generated by resolution from a set of clauses S are made up only of variables and the finite set of predicate symbols and Skolem function symbols occurring in S Therefore if *both* types of growth are limited for all formulas F in a class $\mathbf{C}$ by a complete resolution procedure $\mathbf{R}$, then at some level $p + 1$ of the application of $\mathbf{R}$ to $c(F)$, only variants of clauses in $\mathbf{R}^p(c(F))$ will be generated. Thus $\mathbf{R}^{p+1}(c(F)) \approx \mathbf{R}^p(c(F))$, and $\mathbf{R}$ must be a resolution decision procedure for $\mathbf{C}$.

The strategy of the proofs that the resolution procedures $\mathbf{R}_2$ and $\mathbf{R}_3$ below are decision procedures for various classes consists of showing that bounds on growth in nesting and growth in size can be computed from the formulas F to which the resolution methods are applied. In most cases these bounds are constant over a class or are based on simple syntactic properties of the formulas. Note that this does not mean that these bounds need to be computed in order to apply these decision procedures to formulas; their existence is used only in the proofs that the procedures will always halt when applied to sets of clauses $c(F)$.

One might suspect that if a bound on only one of growth in nesting or growth in size could be shown to hold for a complete resolution procedure $\mathbf{R}$ with respect to a class $\mathbf{C}$, then $\mathbf{C}$ would be solvable (even though $\mathbf{R}$ might not be a decision procedure for it). However, this is not the case; below we exhibit unsolvable classes for which finite nesting limits and finite size limits exist with respect to complete resolution procedures.

6. *A-Orderings*

In defining the resolution decision procedures below, we use two techniques to attempt to limit the two types of growth mentioned in Section 5. To provide a nesting restriction, the *A-ordering* strategy of Kowalski and Hayes [15] is used; a particular A-order which limits growth in nesting for formulas in several solvable classes is defined. A new technique of simplifying clauses, called *condensing*, will be defined in the next section as an attempt to limit clause size.

An *A-ordering* $<_a$ is an irreflexive and transitive binary relation on atomic formulas such that (a) for any set of clauses, $<_a$ is compatible with some enumeration of its Herbrand base (this means that there is some enumeration $A_1, A_2, \cdots$ of the base such that if $A_i <_a A_j$, then $i < j$); and (b) if $A <_a B$, then $A\theta <_a B\theta$ for any substitution θ.

(In their original definition, Kowalski and Hayes omit (a), though it is necessary to their proof For example, if a set of clauses contains only the monadic predicate letters P and Q, the ordering $<_q$ defined by $P(t) <_q Q(u)$ for all terms t and u obeys (b) but not (a) for an infinite Herbrand universe.)

If $<_a$ is a binary relation on atoms, a resolvent E of clauses C and D with resolved atom L will be said to *obey the* $<_a$ *restriction* if there is no atom M occurring in E such that $L <_a M$. The resolution procedure $\mathbf{R}_{<_a}$ corresponding to $<_a$ is defined by:

$$\mathbf{R}^0_{<_a}(S) = S,\ \mathbf{R}_{<_a}(S) = S \cup \{E \mid E \text{ is a resolvent of } C, D \in S \text{ obeying the } <_a\text{-restriction}\}.$$

Kowalski and Hayes show that $\mathbf{R}_{<_a}$ for an A-ordering $<_a$ is complete via semantic trees by using the enumeration compatible with $<_a$ (see (a) above) to generate the tree.

Often, the functional *height* of terms occurring in atoms is used in constructing enumerations for semantic trees. Here we let the height of a constant or variable be 1, the height of a term $f(t_1, \cdots, t_n)$ be one more than the maximum of the heights of $t_1, \cdots, t_n$, and the height of an atom be the maximum of the heights of its arguments.

We now define a particular A-ordering designed to restrict growth in functional nesting. This ordering is motivated by a desire to remove atoms with higher functional nesting first (by resolving upon them) before the nesting increases further because of substitution. First, an ordering $<_1$ on terms is defined.

We say terms r and s are *similar* if they differ only in their (outermost) function symbols. Term r will be said to *dominate* term s if $r = f(t_1, \cdots, t_n)$, $s = g(t_1, \cdots, t_m)$, and $0 \le m < n$. Then $r <_1 s$ if either (1) r occurs in s, (2) r is dominated by s, or (3) a term u which dominates r or is similar to r occurs in s.

LEMMA 6.1 *Relation $<_1$ on terms is irreflexive, transitive, and preserved under substitution.*

PROOF. $<_1$ is clearly irreflexive, since a term cannot occur in itself, cannot be dominated by itself, and cannot be dominated by or be similar to a term which occurs in itself. $<_1$ is preserved under substitution, since if $r <_1 s$ by (1), (2), or (3), then $r\theta <_1 s\theta$ by (1), (2), or (3), respectively, for any substitution θ. $<_1$ is transitive, since consideration of the various cases shows that:

if $r <_1 s$ by (1) and $s <_1 t$ by (1), (2), or (3), then $r <_1 t$ by (1);
if $r <_1 s$ by (2) and $s <_1 t$ by (2), then $r <_1 t$ by (2);
if $r <_1 s$ by (2) and $s <_1 t$ by (1) or (3), then $r <_1 t$ by (3);
if $r <_1 s$ by (3) and $s <_1 t$ by (1), (2), or (3), then $r <_1 t$ by (3). ∎

$<_2$ is defined on atoms A and B by letting $A <_2 B$ if and only if there is some argument u of B such that for each argument t of A, $t <_1 u$.

THEOREM 6.2. *$<_2$ is an A-ordering.*

PROOF. The irreflexivity and preservation under substitution of $<_2$ follow from the same properties of $<_1$. For transitivity, let $A <_2 B$ and $B <_2 C$, and let t be any argument of atom A. Then since $A <_2 B$, there is some argument u of B such that $t <_1 u$. But since $B <_2 C$, there must be some argument v of C such that $u <_1 v$, and by transitivity of $<_1$, $t <_1 v$. Hence $A <_2 C$.

To show that there is an enumeration of the Herbrand base of any set of clauses compatible with $<_2$ in the sense of (a) above, we note that if $A <_2 B$ for ground atoms A and B, then the height of A must be less than or equal to the height of B. Thus the enumeration of the Herbrand base, in which atoms smaller in height precede those greater in height, and in which the finite sets of atoms of the same height are ordered by $<_2$ if possible and otherwise lexicographically (which is possible because of the irreflexivity and transitivity of $<_2$), is the desired enumeration. ∎

Examples. $a <_1 f(x)$, $x <_1 f(x)$, $f(x) <_1 g(x, y)$, $P(x, f(x)) <_2 Q(g(x, y))$.

Given clauses $C = \{P(x, f(x)), Q(g(x, y))\}$ and $D = \{P(u, v), \neg Q(w)\}$, their resolvent $E = \{P(x, f(x)), P(u, v)\}$ obeys the $<_2$ restriction, but resolvent $F = \{Q(g(x, y)), Q(w)\}$ does not.

7 *Condensing*

In addition to restrictions on growth in nesting (via A-ordering and some modifications of it), limits on growth in clause size must be provided to assure a decision procedure. To do this we introduce the notion of *condensing*, in which we try to collapse as many literals in a clause as possible by a substitution.

We define the *condensation* of a clause to be the smallest subset of the clause which is also an instance of it. Though one can effectively determine for any subset of a clause whether it is an instance, it is not obvious that a unique condensation always exists But uniqueness up to variants can be established:

THEOREM 7.1 *Any two condensations of a clause are variants.*

PROOF Let $A\theta_1 \subseteq A$ and $A\theta_2 \subseteq A$ be condensations of A. Now $A\theta_1\theta_2 \subseteq A\theta_2 \subseteq A$. But $A\theta_1\theta_2$ cannot have fewer literals than $A\theta_2$, for then $A\theta_2$ would not be a condensation. So $A\theta_1\theta_2 = A\theta_2$. Similarly $A\theta_2\theta_1 = A\theta_1$. Thus $A\theta_1$ and $A\theta_2$ are variants. ∎

A condensation of a clause may be produced by examining all subsets of it to determine if they are instances, or by successively unifying pairs of literals in a clause until no clause which is a variant of a subset of the clause can be produced. For our purposes it is sufficient to note that there is an algorithm which produces a certain condensation (or a variant of it), which will be called *the* condensation. A clause which is its own condensation will be called *condensed.*

Examples If $C = \{Q(x), Q(f(a))\}$, its condensation is $\{Q(f(a))\}$.
Clause $D = \{P(w, x), P(x, y), P(y, z)\}$ is condensed.

We now define $cond(S)$ for a set of clauses S as the set which results from S when each clause is replaced by its condensation, and use this to define a resolution procedure. If $\mathbf{R}$ is any resolution procedure, $\mathbf{R}_{cond}$ is the procedure which is like $\mathbf{R}$ except that each initial clause and generated resolvent is replaced by its condensation:

$$\mathbf{R}^0_{cond}(S) = cond(\mathbf{R}^0(S)), \qquad \mathbf{R}_{cond}(S) = S \cup cond(\mathbf{R}(S) - S).$$

$\mathbf{R}_{cond}$ is a resolution procedure under our general definition which allows specific instances of resolvents to be formed in computing $\mathbf{R}_{cond}(S)$.

Condensation is compatible with certain resolution procedures in the following sense:

THEOREM 7.2. *If* $\mathbf{R}$ *is a resolution procedure which is complete via semantic trees, then* $\mathbf{R}_{cond}$ *is complete.*

PROOF. First, note that if $\mathbf{R}^0(S)$ is unsatisfiable, then $\mathbf{R}^0_{cond} = cond(\mathbf{R}^0(S))$ is unsatisfiable since $cond(\mathbf{R}^0(S))$ contains a subset of every member of $\mathbf{R}^0(S)$. If $\mathbf{R}$ is complete via semantic trees, then there is a semantic tree T for any set of clauses such that if C and D fail immediately below a node w of T, but no higher, then some resolvent of C and D produced by $\mathbf{R}$ fails at w. But by the definition of failure, the condensations C' and D' of C and D must also fail below w, since they are subsets of C and D. Therefore by the completeness of $\mathbf{R}$ via semantic trees, some resolvent E of C' and D' produced by $\mathbf{R}$ must fail at w. But the condensation E' of E, produced by $\mathbf{R}_{cond}$, also fails at w. Thus $\mathbf{R}_{cond}$ is complete via semantic trees, and hence complete. ∎

In Section 8 we use both A-ordering and condensation to produce a resolution decision procedure.

8. *Resolution Decision Procedure* $\mathbf{R}_2$

In this section we consider the resolution procedure $\mathbf{R}_2 = \mathbf{R}_{<_i,cond}$, which is complete by the results of Sections 6 and 7 This procedure is shown below to be a decision procedure for some of the "simpler" solvable classes: the monadic predicate calculus and the Ackermann class (prefix $\exists^*\forall\exists^*$) In proving this we show that for each formula F in such classes, there exist

(1) A *nesting limit* g such that no clause in $\mathbf{R}_2^p(c(F))$ for any $p \geq 0$ has terms of height greater than g, and (2) a *size limit* s such that no clause in $\mathbf{R}_2^p(c(F))$ for any $p \geq 0$ has more than s literals.

If such limits can be computed for each formula F in a class $\mathbf{C}$, then $\mathbf{R}_2$ is a decision procedure for $\mathbf{C}$: Since the literals in each level $\mathbf{R}_2^p(c(F))$ are made up only of variables and the finite set of predicate and function symbols in $c(F)$, only a finite number of nonvariant clauses exist within the limits g and s. Therefore if F is satisfiable, there must be a $p \geq 0$ such that all clauses generated after level p are variants of previously generated clauses. Then $\mathbf{R}_2^p(c(F)) \approx \mathbf{R}_2^{p+1}(c(F))$, $\mathbf{R}$ will halt without generating $\Box$, and F will be shown satisfiable.

Note carefully that this does not mean that nesting or size limits need be the same for all formulas in a class, nor that the limits must be computed for each formula to which $\mathbf{R}_2$ is to be applied. It is only necessary that we show that such limits exist. Having shown this, we may then apply $\mathbf{R}_2$ to any satisfiable formula in a class $\mathbf{C}$ and be assured that it will halt at some level. (And since $\mathbf{R}_2$ is complete, we are assured that it will produce $\Box$ given any unsatisfiable formula.)

The nesting limit which we shall show for each of the classes discussed below is two. That is, we show that *no* nesting of function symbols can take place; the only arguments to function symbols will be variables and constants (Note that this is true of the initial clauses because these are classes of pure formulas; the only functions appearing in the clause representation are Skolem functions.) To show that a size limit exists given this nesting limit, we prove that only a finite number of condensed, nonvariant clauses with height of literals two or less can be formed by $\mathbf{R}_2$.

To aid in establishing a size limit, clauses are partitioned (for purposes of the proof only) into blocks with common variables. The *v-partition* of a clause is a partition such that (a) all the ground literals are in a single block (called the *ground block*) containing only ground literals; (b) no two blocks share variables; (c) no refinement of the partition obeys (a) and (b) Given a nesting limit, a size limit for condensed clauses may be established by showing that the size of a block of the v-partition is bounded. Then any clause having more than some number of blocks must have two blocks which are variants, and will not be condensed.

Our technique in proving that a class $\mathbf{C}$ has nesting and size limits with respect to $\mathbf{R}_2$ (or the procedure $\mathbf{R}_3$ introduced in Section 9) involves defining certain properties of clauses and proving things about them. A property $\mathbf{K}$ of clauses is an *acceptable property* if it obeys these conditions:

(a) If $\mathbf{K}$ holds for a clause, it holds for any variant or subset.

(b) If $\mathbf{K}$ holds for two clauses sharing no variables, it holds for their union.

(c) Given a finite set of predicate and function symbols, only a finite number of nonvariant condensed clauses obeying $\mathbf{K}$ can be constructed.

To show that $\mathbf{R}_2$ is a decision procedure for a class **C**, then, we need to:

(1) Define an acceptable property **K**.

(2) Prove that all clauses in $c(F)$ for any formula **F** in **C** obey **K**.

(3) Prove that any resolvent of clauses obeying **K** which is permitted by $\mathbf{R}_2$ also obeys **K**.

Part (3) can be simplified even further A resolvent is a subset of an instance (under the unifying substitution) of the union of variants of two clauses. Moreover, the unifying substitution may be obtained by successively unifying pairs of atoms. Note also that if $L <_2 M$, then L and M cannot unify, since this would contradict the irreflexivity of $<_2$ Therefore, considering these facts, the definition of acceptable property, and the notion of A-ordering, it is sufficient to replace (3) by·

(3′) Prove that if clause C with literals L and M obeys **K**, and L and M unify with most general unifier θ such that there is no atom N in **C** such that $L\theta = M\theta <_2 N\theta$, then $C\theta$ obeys **K**.

To illustrate the way in which these demonstrations are carried out, one proof that $\mathbf{R}_2$ is a decision procedure for a class will be given in detail. For the other solvable classes, the acceptable property for that class will be defined, with details of the proof (which appear in [11]) left to the reader.

THEOREM 8.1 $\mathbf{R}_2$ *is a resolution decision procedure for the Ackermann class* $\exists^*\forall\exists^*$.

PROOF. Let F be a formula with prefix $\exists^m\forall\exists^n$, and let $a_1, \cdots, a_m$ be the constants and $f_1, \cdots, f_n$ be the monadic function symbols occurring in $c(F)$.

The property ACK which we define for clauses coming from this class is that (i) no literal of height greater than two appears; (ii) exactly one variable occurs in each nonground block of the v-partition; and (iii) each argument of each literal in a nonground block with variable v is in $\{a_1, \cdots, a_m, v, f_1(v), \cdots, f_n(v)\}$.

Inspection shows that ACK is acceptable. It provides a nesting limit of two, and provides a size limit because the number of literals which can appear in a single block of the v-partition of a clause obeying ACK is the finite number of literals constructible from k predicate symbols, $m + n$ function symbols, and one variable. This is given by $2\sum_{i=1}^{k} (m + n + 1)^{\deg(P_i)} \leq 2k(m + n + 1)^q$, where $\deg(P_i)$ is the degree of the ith predicate symbol and q is the maximum degree of the predicate symbols appearing

Now, since only one universal quantifier appears in the prefix of F, only one variable can appear in $c(F)$. So all clauses of $c(F)$ must obey ACK Moreover, since only monadic function symbols appear, no more than one variable can *ever* occur in a literal (although by standardization apart and unification, *clauses* with more than one variable can be produced). Also, if a term $f_i(v)$ occurs as an argument to a predicate symbol in a literal (which must be an instance of a literal in $c(F)$), no term of the form $f_j(a_k)$ can also occur. Therefore all clauses produced by $\mathbf{R}_2$ from $c(F)$ obey (ii) and (iii) of ACK. Now it is only necessary to show that, consistent with $<_2$, no unification of two atoms of a clause obeying ACK can increase functional nesting.

Suppose that in the unification of L and M by substitution σ in clause C obeying ACK, growth in nesting were to occur. Then one literal, say L, would have to have a variable v as argument, the corresponding term of M would have to be of the form $f_j(t)$, t another variable or constant, and a term $f_i(v)$ would have to appear either in L or some other atom. But if $f_i(v)$ did occur in another atom N in C, then some term $f_k(v)$ would have to be an argument to L. Otherwise, because C obeys ACK, $L <_2 N$ and $L\sigma <_2 N\sigma$, violating the restriction of $<_2$ imposed in (3′). So L must have an argument $f_k(v)$ But then the unification of L and M cannot take place, because the only possibilities for the corresponding argument position in M (indicated by ? in Figure 1) are t, some $f_p(t)$, or a constant, and none of these can unify with $f_k(f_j(t))$. Thus no nesting can be produced. ∎

The monadic class is also decided by $\mathbf{R}_2$. The argument for a limit to clause size is

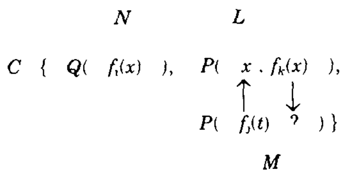

FIG 1

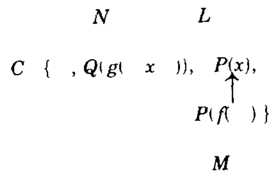

FIG 2

slightly more difficult, but the establishment of the nesting limit is similar to Theorem 8.1.

THEOREM 8.2 $\mathbf{R}_2$ *is a resolution decision procedure for the monadic class.*

This theorem is proved using the acceptable property *MON*.

MON is defined for clauses from a formula F with n universal quantifiers as follows: (i) no literal of height greater than two appears, (ii) each block of the v-partition has at most n variables; and (iii) the variables of each nonground block can be listed $v_1, \cdots, v_m$, $m \leq n$, such that the arguments of all literals of the block are either constants, among the $v_1, \cdots, v_m$, or of the form $f(v_1, \cdots, v_j)$, $1 \leq j \leq m$ (i.e. the list of arguments of each function symbol is an initial segment of $v_1, \cdots, v_m$).

If growth in nesting were to occur, the situation shown in Figure 2 would have to obtain; as in the case shown in Figure 1, this situation violates the conditions imposed by $<_2$.

Note that if $\mathbf{R}_2$ is applied to the set of two clauses generated from the satisfiable formula (5 2) in Section 5, which is in both the Ackermann class and the monadic class, the formula F is shown satisfiable.

$\mathbf{R}_2^0(c(F)) = \{\{P(x), P(f(x))\}, \{\neg P(x), \neg P(f(x))\}\}$,
$\mathbf{R}_2^1(c(F)) = \mathbf{R}_2^0(c(F)) \cup \{\{P(x), \neg P(x))\}\}$,
$\mathbf{R}_2^2(c(F)) \approx \mathbf{R}_2^1(c(F))$.

Though $\mathbf{R}_2$ can be shown to decide satisfiability for a few other less-known classes (see [11]), a question which arises at this point is whether $\mathbf{R}_2$ decides the Godel class (prefix $\exists^*\forall\forall\exists^*$), a natural and simple extension of the Ackermann class Unfortunately it does not, as the following example from that class shows.

Let

$$G = \exists z \forall x \forall y \exists w (\neg P(w, z) \;\&\; (Q(x, y) \vee P(x, z)) \;\&\; (\neg Q(x, w) \vee \neg P(w, y))). \quad (8.1)$$

G is a satisfiable formula of the Godel class, and also of the Maslov class, since each conjunct is at most a binary disjunction $c(G)$ is $\{D_1, D_2, D_3\}$, where

$$D_1 = \{\neg P(f(x, y), a)\}, \quad D_2 = \{Q(x, y), P(x, a)\}, \quad D_3 = \{\neg Q(x, f(x, y)), \neg P(f(x, y), y)\}$$

Now for each n, $\mathbf{R}_2$ admits a derivation of a clause C_{2n+1} of the form shown in Figure 3

Since an unbounded number of clauses C_{2n+1} will be generated from $c(G)$ by $\mathbf{R}_2$ (and are not variants of earlier clauses), $\mathbf{R}_2$ is not a decision procedure for the Godel class (or its extensions) or for the Maslov class In fact, to avoid the production of this derivation, an A-ordering $<_a$ would have to be such that $P(t, a) <_a Q(t, y)$ or

$$\begin{tikzcd}
D_1 \arrow[dd, dash] & \\
 & D_2 \arrow[dl, dash] \\
C_1 = \{Q(f(w_1,w_2), w_3)\} \arrow[dd, dash] & \\
 & D_3 \arrow[dl, dash] \\
C_2 = \{\neg P(f(f(w_1,w_2), w_3), w_3)\} \arrow[dd, dash] & \\
 & D_2 \arrow[dl, dash] \\
C_3 = \{Q(f(f(w_1,w_2), a), w_3)\} \arrow[dd, dash] & \\
 & D_3 \arrow[dl, dash] \\
\vdots & \\
C_{2n} = \{\neg P(\overbrace{f(f(f}^{n+1 \text{ times}}\;(f(w_1,w_2), a)\;\; a), w_2), w_3)\} \arrow[dd, dash] & \\
 & D_2 \arrow[dl, dash] \\
C_{2n+1} = \{Q(\overbrace{f(f(f}^{n+1 \text{ times}}\;(f(w_1, w_2), a)\;\; a), a), w_3)\} &
\end{tikzcd}$$

FIG 3

$$\begin{tikzcd}
\{Q(f(w_1,w_2),w_3)\} \arrow[dr, dash] & & \{\neg Q(x,f(x,y)), P(f(x,y),y)\} \arrow[dl, dash] \\
 & \{P(f(f(w_1,w_2), y), y)\} &
\end{tikzcd}$$

FIG 4

$P(f(t, y), y) <_a Q(t, f(t, y))$. But by substitutions, the argument sets of the atoms on either side of $<_a$ can be made identical, and $<_a$ must be preserved under substitution. Thus any A-ordering would have to take advantage of the names of predicate symbols or the order of their arguments. But then a similar example, with predicate symbols and order of arguments reversed, would defeat $\mathbf{R}_{<_a}$. So no resolution procedure using only A-ordering and condensing can be a decision procedure for the Godel class or the Maslov class.

9. *Resolution Decision Procedure* $\mathbf{R}_3$

As a first step toward developing a resolution procedure which does decide satisfiability for the Godel class, consider again the failure of $\mathbf{R}_2$ when applied to formula (8.1) above. The problem with this example is that a literal $P(x, a)$ which has only one argument in addition to constants is resolved upon, while a literal $Q(x, y)$ in the same clause which has x among its arguments but also has other nonconstant arguments is not resolved upon. This produces the literal $Q(f(w_1, w_2), w_3)$, which remains in the resolvent. The presence of w_3 as an argument in this literal is used (in Figures 3 and 4) to absorb functional arguments and cause nesting

If literals such as $Q(f(w_1, w_2), w_3)$, in which terms of height greater than one appear along with variables which are not arguments of such terms, could be avoided and completeness could be preserved, a resolution decision procedure for the Godel class (and its extensions) could be produced As example (8.1) shows, this cannot be done via A-ordering alone. We now introduce a resolution procedure which extends the A-ordering technique and, again, we rely for the proof of its completeness on a special ordering of the Herbrand base for a semantic tree.

Here, we will call a literal *essentially monadic* if it has only one argument which is not a constant; it is essentially monadic *on* that argument. A literal is *argumental* if it has some argument $t = f(t_1, \cdots, t_n)$ such that each of its other arguments is t, is similar to t, is one of the $t_1, \cdots, t_n$, or is a constant; the literal is argumental *on* t We will call a literal *simple* if for some sequence $v_1, \cdots, v_n$ of variables and constants, all its arguments are variables, constants, or of the form $f(v_1, \cdots, v_n)$ (i.e. all functional terms are of height at most two and have the same sequence of variables and constants as arguments). Literals which are both simple and essentially monadic or simple and argumental will be called *simply monadic* or *simply argumental*, respectively.

In the above example from the Godel class, resolved atom $P(f(w_1, w_2),a)$ is simply monadic on $f(w_1,w_2)$. Literal $Q(f(w_1w_2),w_3)$ is neither simple, argumental, nor essentially monadic because of the presence of w_3. In the resolution decision procedure described below, which decides the Godel class, each resolvent containing such nonargumental literals is replaced by a finite set of instances of it in which all literals of height greater than one are argumental. We now define the way in which such sets of instances are constructed.

Let K be a set of function symbols, and let E be a resolvent of two clauses C and D obeying the $<_2$ restriction, with resolved atom P We define $\Phi(E,P,K)$ as follows:

(1) If P is simply monadic on t, let Q be the first simple literal in E, in lexicographic order, which has a term u which is t or is similar to t as argument, and which has as arguments variables $v_1, \cdots, v_p$ which do not occur in t (and hence not in P). If no such literal exists, let $\Phi(E,P,K) = \{E\}$. Otherwise, let $C = \{u, c_1, \cdots, c_r\}$, where $c_1, \cdots, c_r$ are the constants in K. Let $\sigma_1, \cdots, \sigma_m$ be all substitutions of the form $\{v_1 \leftarrow u_1, \cdots, v_p \leftarrow u_p\}$, where $u_i \in C$, $1 \le i \le p$, and such that for no literal L in E is $P <_2 L\sigma_i$. Then let $\Phi(E,P,K) = \bigcup_{i=1}^{m} \Phi(E\sigma_i,P,K)$.

(2) If P is simply argumental on term $t = f(t_1, \cdots, t_n)$, but not simply monadic, then let Q be the first simple literal in E in lexicographic order which has t or a similar term as argument but which is not simply argumental. If no such literal exists, let $\Phi(E,P,K) = \{E\}$ Otherwise, let $v_1, \cdots, v_p$ be the variables which are arguments of Q but which do not occur in t. Let

$$C = \{t, t_1, \cdots, t_n, g_1(t_1, \cdots, t_n), \cdots, g_k(t_1, \cdots, t_n), c_1, \cdots, c_r\},$$

where $g_1, \cdots, g_k$ are all the function symbols of K with degree equal to that of f, and $c_1, \cdots, c_r$, are the constants in K. Again, let $\sigma_1, \cdots, \sigma_m$ be all substitutions of the form $\{v_1 \leftarrow u_1, \cdots, v_p \leftarrow u_p\}$, where $u_i \in C$, $1 \le i \le p$, and such that for no literal L in E is $P <_2 L\sigma_i$. Then let $\Phi(E,P,K) = \bigcup_{i=1}^{m}\Phi(E\sigma_i,P,K)$.

(3) Otherwise, let $\Phi(E,P,K) = \{E\}$

Informally, suppose E is a resolvent with resolved atom P which is simply argumental or simply monadic on a term t If E contains simple literals with t as argument which are not simply monadic or simply argumental (as in Figure 3), Φ "makes" them simply argumental or simply monadic by a substitution, attempting to prevent the growth in nesting of example (8.1).

Using Φ, we define the resolution procedure $\mathbf{R}_3$ by: $\mathbf{R}_3^0(S) = cond(S)+$ $\mathbf{R}_3(S) = S \cup cond(S')$, where S' is the union of the sets $\Phi(E,P,K)$ for all resolvents E (with resolved atom P) of clauses C and D in S which obey the $<_2$ restriction, and where K is the set of function symbols occurring in S, augmented by the constant a if no constants appear.

$\mathbf{R}_3$ appears to be a resolution procedure, since our broad definition allows instances of clauses to be formed. All that is necessary is to show that each $\Phi(E,P,K)$ is finite. But since each substitution σ used in constructing $\Phi(E,P,K)$ introduces no new variables and removes some number p of variables from E on each application of Φ, the recursive application of Φ must terminate with a finite set of instances. Thus $\mathbf{R}_3$ is a resolution procedure.

Examples. Consider again the troublesome example (8.1). The resolution of $\{\neg P(f(x, y), a)\}$ and $\{Q(x, y), P(x, a)\}$ under $\mathbf{R}_2$ produced $E = \{Q(f(w_1, w_2), w_3)\}$ with resolved atom $P = P(f(w_1, w_2), a)$. This resolved atom is essentially monadic on $f(w_1, w_2)$. Therefore

$$\Phi(E,P,\{a,f\}) = \{\{Q(f(w_1, w_2),a)\}, \{Q(f(w_1w_2), f(w_1w_2))\}\}.$$

If $E_1 = \{P(w, f(x, y))\}$, $P_1 = Q(f(x, y), y)$, $E_2 = \{P(x, f(x, y))\}$, $P_2 = Q(f(x, y), y)$, then

$\Phi(E_1, P_1, \{f, a\}) = \{\{P(x, f(x, y))\}, \{P(y, f(x, y))\}, \{P(f(x, y), f(x, y))\}, \{P(a, f(x, y))\}\}$,
$\Phi(E_2, P_2, \{f, a\}) = \{E_2\}$.

If $\mathbf{R}_3$ is applied to the set of clauses $c(G)$ of Section 8, then

$\mathbf{R}_3^0(c(G)) = \{D_1, D_2, D_3\}$,
$\mathbf{R}_3^1(c(G)) = \mathbf{R}_3^0(c(G)) \cup \{\{Q(f(x, y), a)\},\{Q(f(a, a), a), \neg Q(a, f(a, a)\}, \{P(x, a), \neg P(f(x, y), y)\}\}$,
$\mathbf{R}_3^2(c(G)) = \mathbf{R}_3^1(c(G)) \cup \{\{P(a, a), Q(f(a, a), a)\}, \{Q(f(x, a), a), P(x, a)\}\}$,
$\mathbf{R}_3^3(c(G)) = \mathbf{R}_3^2(c(G))$.

So $\mathbf{R}_3$ can demonstrate the satisfiability of G where $\mathbf{R}_2$ failed.

THEOREM 9.1. $\mathbf{R}_3$ *is complete*.

PROOF. Recall from Theorem 6.2 the enumeration of the Herbrand base of any set of clauses used to prove that $\mathbf{R}_2$ is an A-ordering. In this enumeration, atoms smaller in height precede those greater in height, and the finite sets of clauses of the same height are ordered by $<_2$. To prove $\mathbf{R}_3$ complete, we impose additional restrictions on this ordering. For atoms of the same height which are incomparable under $<_2$, let essentially monadic atoms precede atoms not essentially monadic, and let argumental (but not essentially monadic) atoms precede nonargumental atoms. Otherwise, let the ordering be lexicographic.

Let T be the semantic tree for any set of clauses S based on this enumeration, and let C and D be clauses failing immediately below, but not at, a node w. From Theorem 6.2, there must exist a resolvent E of C and D (with resolved atom P) which obeys the $<_2$ restriction and which fails at w. We now wish to show that if K is the set of function symbols occurring in S (perhaps augmented by a), then some clause in $\Phi(E,P,K)$ fails at w. Clearly this is true in the cases where $\Phi(E,P,K) = \{E\}$; the crucial cases are (1) and (2) of the definition of Φ in which the substitutions $\sigma_1, \cdots, \sigma_m$ are generated. Since Φ is recursive, we can prove this by showing that if E is a clause and P an atom such that no atom L in E is such that $L <_2 P$, then some clause $E\sigma_i$, $1 \le i \le m$, fails at w By induction on the number of applications of Φ, some clause in $\Phi(E,P,K)$ must then fail at w.

Suppose P is essentially monadic on a term t, and there is a simple literal Q in E which has a term u which is t or similar to t as argument, as well as variables $v_1, \cdots, v_p$ which do not occur in t. Let θ be the substitution such that $E\theta$ is contradicted by the assignment at w. Atom $P\theta$ must be essentially monadic on $t\theta$. By the construction of T, $Q\theta$ must be essentially monadic on $u\theta$. Therefore the substitution θ must replace each variable $v_1, \cdots, v_p$ by $u\theta$ or by a constant. Then $\theta = \sigma\theta$, where $\sigma = \{v_1 \leftarrow u_1, \cdots, v_p \leftarrow u_p\}$, with each u_i, $1 \le i \le p$, being u or a constant. But then $E\sigma$ fails at w, and σ is one of the substitutions $\sigma_1, \cdots, \sigma_m$

If P is argumental on term $t = f(t_1, \cdots, t_n)$, let Q be the first simple literal in E which is argumental on t or a similar term, and which has variables $v_1, \cdots, v_p$ which do not occur in t as arguments. Let θ be as above. Then $P\theta$ must be argumental on $t\theta$, and by the construction of T, $Q\theta$ must be argumental on $t\theta$ or a similar term. Then θ must replace each of $v_1, \cdots, v_p$ by a term similar to $t\theta$, an argument of $t\theta$, or a constant. But then (as above), $\theta = \sigma\theta$, where σ replaces each such variable by t, a term similar to t, an argument of t, or a constant, respectively. But then $E\sigma$ fails at w, and σ is one of the substitutions $\sigma_1, \cdots, \sigma_m$ used in constructing Φ.

Thus in all cases, if E is a resolvent with resolved atom P obeying the $<_2$ restriction which fails at w in T, then some clause in $\Phi(E,P,K)$ also fails at w. ∎

Although $\mathbf{R}_3$ is not a refinement of $\mathbf{R}_2$, in the sense that it generates a subset of the clauses generated by $\mathbf{R}_2$, it does decide satisfiability for the classes considered in Section 8. This is seen by noting that if a resolvent E with resolved atom P obeys the $<_2$-restriction and property *ACK* or property *MON*, then $\Phi(E,P,K) = \{E\}$. Thus for these classes, $\mathbf{R}_3$ generates the same clauses as $\mathbf{R}_2$.

We now show that $\mathbf{R}_3$ is more powerful than $\mathbf{R}_2$, that is, that it decides satisfiability for the Godel class and some of its extensions. We do this by first showing that when $\mathbf{R}_3$ is applied to *any* formula with prefix $\exists^*\forall^*\exists^*$, no growth in nesting can occur. Since the class $\exists^*\forall^*\exists^*$ is unsolvable, no size limit can exist, so when $\mathbf{R}_3$ is applied to some sets of clauses in this class, unbounded growth in clause size must occur. But for the Maslov class and the extended Skolem class, which are solvable subcases of $\exists^*\forall^*\exists^*$, the additional constraints imposed on the formulas make a size limit possible.

THEOREM 9.2. *If F is a pure formula with prefix $\exists^*\forall^*\exists^*$, then no clause in $\mathbf{R}_3^p(c(F))$, $p \geq 0$, has any literal of height greater than two*

PROOF. The technique used to prove this theorem is that used in Section 8 with $\mathbf{R}_2$. A property of clauses is defined, and it is shown that the initial clauses and all resolvents generated by $\mathbf{R}_3$ have this property. Of course, the property used in this theorem cannot be acceptable (because it cannot provide a size limit), but it does give a nesting limit of two for the $\exists^*\forall^*\exists^*$ class.

The property *SKO* used is that (i) no literal in a clause has height greater than two, and (ii) every literal of height greater than one is simply argumental.

Clearly every clause in $c(F)$ obeys *SKO* To show that clauses produced by $\mathbf{R}_3$ also have this property, we first note that if clauses C and D obey *SKO*, then no literal in a resolvent E of C and D with resolved atom P which obeys the $<_2$ restriction can have any literal of height greater than two. However, E can have literals of height two which are not simply argumental; i.e. there may be a first literal Q in which a term t of height two, and variables not occurring in t as arguments, occur Since this violation of *SKO* was created by the substitution which produced resolved atom P, by a substitution of t for a variable, the terms in Q of height two must be t or terms similar to t But by the construction of Φ, such terms are made simply argumental (or perhaps essentially monadic) in the instances of E in $\Phi(E,P,K)$. Also, the substitutions which generate these instances cannot cause growth in nesting, for then there would be a literal L in some clause in $\Phi(E,P,K)$ such that $P <_2 L$, and this is not allowed by the construction. Hence all clauses produced by $\mathbf{R}_3$ obey *SKO*. ∎

This class and this theorem show that establishment of a nesting limit alone is not sufficient to insure solvability.

COROLLARY 9.3. *$\mathbf{R}_3$ is a resolution decision procedure for the extended Skolem class*

We sketch the proof by describing a property *SKO*2 which provides a size limit for the extended Skolem class; in conjunction with Theorem 9.2, this assures a decision procedure.

The property *SKO*2 is that (i) each nonground block of the v-partition of a clause in which a literal simply argumental on a term $f(v_1, \cdots, v_n)$ occurs is such that every other literal of the block is simply argumental on a similar term, or essentially monadic, or of height one with variables among $v_1, \cdots, v_{n_0}$, where n_0 is as in the definition of this class, and (ii) each other nonground block in which more than one variable appears is such that for some set of m variables, $1 \leq m \leq n_0$, each literal has among its arguments all of these variables, or is essentially monadic on one of them.

The Maslov class has the prefix $\exists^*\forall^*\exists^*$, so Theorem 9.2 provides a nesting limit for it But since formulas in this class have only binary disjunctions in their conjunctive normal forms, clauses produced from them have at most two literals. Thus the class has an inherent size limit of two, for no larger resolvents can be produced. Even without the use of condensing, partitioning, or other arguments, we get the solvability of this class:

COROLLARY 9.4. $\mathbf{R}_3$ *is a decision procedure for the Maslov class.*

The class of all formulas with Krom matrix (conjunction of binary disjunctions) is unsolvable. Since this class has an inherent size limit of two, existence of a size limit alone does not guarantee solvability; some formulas must produce unbounded nesting growth It is the interaction of matrix structure (Krom matrix) and prefix ($\exists^*\forall^*\exists^*$) that causes the solvability of the Maslov class.

10. *Model Construction*

In [22], Luckham and Nilsson describe a way of extracting information (a proof, in some form) from a resolution derivation of the empty clause If a resolution decision procedure indicates that a formula is satisfiable, it would be helpful to obtain analogous information. In this section we describe a way in which a *model,* or satisfying truth assignment to all ground literals, can be constructed when a resolution method halts without producing $\square$. Production of such a model is inherent in other decision methods (such as the amenability method [8]), which work by constructing a truth assignment.

When a resolution procedure halts without producing the empty clause from a formula F, a model for the original formula is not immediately at hand. What is available is the finite set of clauses $\mathbf{R}^p(c(F)) \approx \mathbf{R}^{p+1}(c(F))$ from which no new resolvents can be formed. We use this set below to specify a model. Our only requirement for a resolution procedure used below is that it be proved complete using semantic trees. This requirement is satisfied by the resolution decision procedures $\mathbf{R}_2$ and $\mathbf{R}_3$, and many other resolution strategies.

THEOREM 10.1. *If* $\mathbf{R}$ *is a resolution procedure which is complete via semantic trees and for some formula* F, $\mathbf{R}^p(c(F)) \approx \mathbf{R}^{p+1}(c(F))$ *and* $\square \notin \mathbf{R}^p(c(F))$ *for some* $p \geq 0$, *then a model for* F *can be effectively specified*

PROOF. Let T be the semantic tree for $c(F)$ based on the enumeration for which $\mathbf{R}$ is complete via semantic trees. Recall from Section 3 that if $c(F)$ is satisfiable, then T is not closed for $c(F)$. Therefore there must exist at least one branch from the root on which there are no failure nodes, and the literals on such a branch determine a model for $c(F)$. Also, T can have no inference nodes with respect to $\mathbf{R}^p(c(F))$; otherwise a new resolvent could be formed at that node, and it would not be the case that $\mathbf{R}^p(c(F)) \approx \mathbf{R}^{p+1}(c(F))$.

We wish to use these facts to "walk" down a specific open branch of T. We begin at the root, and when we are at any node, we move to the son of that node which is *not* a failure node. There must be at least one such son, because no inference nodes exist. If neither son is a failure node, we take the left arbitrarily. We can effectively determine whether any clause fails at a node w by generating the finite number of ground instances of clauses in $\mathbf{R}^p(c(F))$ in which all literals are no greater in height than any literal in the assignment at w, and then testing whether this finite assignment contradicts any such instance.

This procedure allows us to walk arbitrarily far down an open, and in general infinite, branch of T, which determines a model for $\mathbf{R}^p(c(F))$, and hence for $c(F)$ and for F. For an infinite model to be effectively specified, we must be able to decide for any ground atom Q over the Herbrand universe of F whether it is true or false in the model. To do this we begin walking along the open branch from the root. Eventually we must walk along an edge labeled by either Q or $\neg Q$; we then stop and assign Q true or false, respectively. ∎

11 *Conclusions*

Two resolution procedures $\mathbf{R}_2$ and $\mathbf{R}_3$ have been shown to be complete in the general case, and to be decision procedures for successively wider classes of pure first-order formulas. $\mathbf{R}_3$, in fact, decides satisfiability for all of the solvable classes in Table I except the Bernays-Schonfinkel class (prefix $\exists^*\forall^*$). The failure for the Bernays-Schonfinkel class is somewhat surprising, since the class is one of the "simpler" solvable classes, easily shown solvable by the finiteness of its Herbrand universe But while $\mathbf{R}_3$, or indeed any resolution procedure, provides a nesting limit for this class, clauses may increase in size without limit, as in example (5.3). The solvability of this class seems to rest strongly on the finiteness of the universe, and not to be related to the considerations which show the other classes solvable via resolution.

$\mathbf{R}_3$, the strongest decision procedure produced here, also fails on other solvable classes not mentioned above, including the Aanderaa class ($\forall\exists\forall$ prefix and Krom matrix) [1] and the Surányi extensions to the extended Skolem class [8]. What $\mathbf{R}_2$ and $\mathbf{R}_3$ do show is that the resolution method, designed for refutation, can be modified to show satisfiability in some cases. Unlike some earlier decision procedures [2], these methods are complete in the general case and do not require a formula in a special form (other than the form of resolution in general) They are applied uniformly to all formulas, and are guaranteed to halt for all unsatisfiable formulas and satisfiable formulas in certain classes.

The operation of $\mathbf{R}_2$ and $\mathbf{R}_3$ on the solvable classes studied gives some insight into the reasons for their solvability. In particular, in the Maslov case, the interaction between the Krom matrix and the Skolem prefix to produce solvability (where neither alone is sufficient) is clarified in the use of $\mathbf{R}_3$. The use of resolution in this way indicates its possibilities as a logical tool, in addition to being a machine-implementable inference mechanism. In particular, resolution might be used to investigate the solvability of classes with open decision problems. Chief among these are classes of the predicate calculus with equality; techniques such as paramodulation [27] might be used to attack this problem.

The obvious advantage in using a complete resolution decision procedure (aside from any efficiency considerations) is that, especially in question-answering applications, negative answers to questions of unsatisfiability can be given more often. $\mathbf{R}_3$ will halt when applied to many formulas on which previous resolution strategies would not. Decision procedures are also of value in proving properties about abstract programs. Manna [23] converts the problem of proving termination of programs to one of showing a first-order formula unsatisfiable, and points out the equivalence in this sense of certain classes of abstract programs and the three solvable classes determined by prefix alone. In [18], Lewis describes in greater detail reductions of assertions about programs to first-order formulas in solvable classes.

Factors detrimental to the efficiency of $\mathbf{R}_2$ and $\mathbf{R}_3$ are the necessity to condense clauses, to detect variants of clauses, to compute $<_2$ between literals, and to compute the set of clauses $\Phi(E,P,K)$. Though no attempt has been made here to consider efficiency, we note that computation of Φ, which seems to involve the generation of several clauses from each resolvent, might be implemented instead by returning the original resolvent, with restrictions on what unifying substitutions can be applied to it. Condensing itself is less costly than the deletion of subsumed clauses, for it is a local rather than a global operation. (Deletion of subsumed clauses might itself be used to strengthen resolution decision procedures. Proofs of this would likewise involve considering an entire set of clauses, and would probably be more difficult than those above)

Theorem 9.3, which states that $\mathbf{R}_3$ produces no nesting growth on formulas with prefix $\exists^*\forall^*\exists^*$, is of interest from the standpoint of the implementation of resolution theorem provers. It follows from this that a resolution procedure like $\mathbf{R}_1$ (the unrestricted original procedure) that discards all resolvents in which growth in nesting occurs is complete on the prefix class $\exists^*\forall^*\exists^*$. Also, *any* first-order formula has a corresponding form with prefix $\forall^*\exists^*$ which is satisfiable if and only if the formula is satisfiable. (This was an early result of Skolem; the transformation of a formula to this form, which is not difficult, is detailed in [6]) Thus a complete resolution technique would be to transform any formula to this Skolem normal form, and to apply unrestricted resolution to the result, discarding all clauses in which any nesting occurs (which is easy to detect). The fact that only limited substitutions of terms for variables can occur in such a method indicates that simpler ways of storing clauses and handling substitution might be used. This, in addition to the reduction in the number of clauses produced, might make such a method a feasible complete resolution strategy.

ACKNOWLEDGMENTS The author wishes to thank Harry R. Lewis, William A. Woods, and the referees for their very helpful comments.

REFERENCES

(Note References [12, 33] are not cited in the text)

1 AANDERAA, S O , AND LEWIS, H R Linear sampling and the $\forall\exists\forall$ case of the decision problem *J Symbolic Logic 39*, 3 (Sept 1974), 519-548
2 ACKERMANN, W *Solvable Cases of the Decision Problem* North-Holland Pub Co , Amsterdam, 1954
3 ANDREWS, P B Resolution with merging *J ACM 15*, 3 (July 1968), 367-381
4 CHANG, C L , AND LEE, R C T *Symbolic Logic and Mechanical Theorem Proving* Academic Press, New York, 1973.
5 CHURCH, A A note on the Entscheidungsproblem *J Symbolic Logic 1* (1936), 40-41, 101-102
6 CHURCH, A. *Introduction to Mathematical Logic, Vol I* Princeton U Press, Princeton, N J , 1956
7 DREBEN, B Solvable Surányi subclasses: An introduction to the Herbrand theory Annals of the Computation Lab of Harvard University, Vol XXXI, Harvard U Press, Cambridge, Mass , 1962, pp 32-47
8 DREBEN, B , AND GOLDFARB, W D *A Systematic Treatment of the Decision Problem* (forthcoming)
9 FRIEDMAN, J A semi-decision procedure for the functional calculus *J ACM 10*, 1 (Jan 1963), 1-24
10 HERBRAND, J. Recherches sur la théorie de la demonstration Thesis, U of Paris, Paris, 1930, partially translated in [33]
11 JOYNER, W H Automatic theorem-proving and the decision problem Tech Rep 7-73, Center for Research in Computing Technology, Harvard U , Cambridge, Mass , 1973
12 JOYNER, W H Automatic theorem proving and the decision problem, 14th Annual Symp on Switching and Automata Theory, Iowa City, Iowa, 1973, 159-166
13 KAHR, A S , MOORE, E F , AND WANG, H Entscheidungsproblem reduced to the *AEA* case *Proc Nat Acad Sci 48* (1962), 365-377
14. KALLICK, B A decision procedure based on the resolution method Proc IFIP Cong 1968, North-Holland Pub Co , Amsterdam, pp 269-275
15 KOWALSKI, R , AND HAYES, P J Semantic trees in automatic theorem-proving In *Machine Intelligence 4*, B Meltzer and D Michie, Eds , Edinburgh U Press, Edinburgh, 1969, pp 87-101
16 KOWALSKI, R , AND KUEHNER, D Linear resolution with selection function *Artificial Intelligence 2* (1971), 227-260
17 KROM, M R The decision problem for formulas in prenex conjunctive normal form with binary disjunctions *J Symbolic Logic 35* (1970), 210-216
18 LEWIS, H R Program schemata and the first-order decision problem *J Comput Syst Scis 8*, 1 (Feb 1974), 71-83
19 LOVELAND, D A linear format for resolution Proc IRIA Symp on Automatic Demonstration, Lecture Notes in Mathematics No 125, Springer-Verlag, New York, 1970, pp 147-162
20 LOWENHEIM, L Uber Moglichkeiten im Relativkalkul *Mathematische Annalen 76* (1915), 447-470 Translated in [33]
21 LUCKHAM, D Refinement theorems in resolution theory Proc. IRIA Symp on Automatic Demonstration, Lecture Notes in Mathematics No 125, Springer-Verlag, New York, 1970, pp 163-190

22 LUCKHAM, D , AND NILSSON, N J Extractıng ınformatıon from resolutıon proof trees *Artıficıal Intellıgence 2* (1971), 27-54

23 MANNA, Z Propertıes of programs and the first-order predıcate calculus *J ACM 16*, 2 (Aprıl 1969), 244-255

24 MASLOV, S JU An ınverse method of establıshıng deducıbılıty ın the classıcal predıcate calculus *Sovıet Math -Doklady 5* (1964), 1420-1424 (translated)

25 NILSSON, N J *Problem-Solvıng Methods ın Artıficıal Intellıgence* McGraw-Hill, New York, 1971

26 OREVKOV, V P Two undecıdable classes of formulas ın classıcal predıcate calculus. Semınars ın Math , Vol 16, V A Steklov Inst , Lenıngrad, 1969, pp 98-102 (translated)

27 ROBINSON, G , AND WOS, L Paramodulatıon and theorem-provıng ın first order theorıes wıth equalıty *Machıne Intellıgence 4*, B Meltzer and D Mıchıe, Eds , Edınburgh U Press, Edınburgh, 1969, pp 135-150

28 ROBINSON, J A A machıne-orıented logıc based on the resolution prıncıple *J ACM 12*, 1 (Jan 1965), 23-41

29 ROBINSON, J A Automatıc deductıon wıth hyperresolutıon *Int J Comput Math 1* (1965), 227-234

30 ROBINSON, J A The generalızed resolutıon prıncıple In *Machıne Intellıgence 3*, D Mıchıe, Ed , Edınburgh U Press, Edınburgh, 1968, pp 77-93

31 SURÁNYI, J Zur Reduktıon des Entscheıdungsproblem des logıschen Funktıonenkalkuls *Matematıkaı es Fızıkaı Lapok 50* (1943), 51-74

32 TURING, A M On computable numbers, wıth an applıcatıon to the Entscheıdungsproblem *Proc London Math Soc 42* (1937), 230-265, *43* (1937), 544-546

33 VAN HEIJENOORT, J , Ed *From Frege to Godel A Source Book ın Mathematıcal Logıc 1879-1931* Harvard U Press, Cambrıdge, Mass , 1967

RECEIVED SEPTEMBER 1975, REVISED FEBRUARY 1976