R-803-PR
August 1971

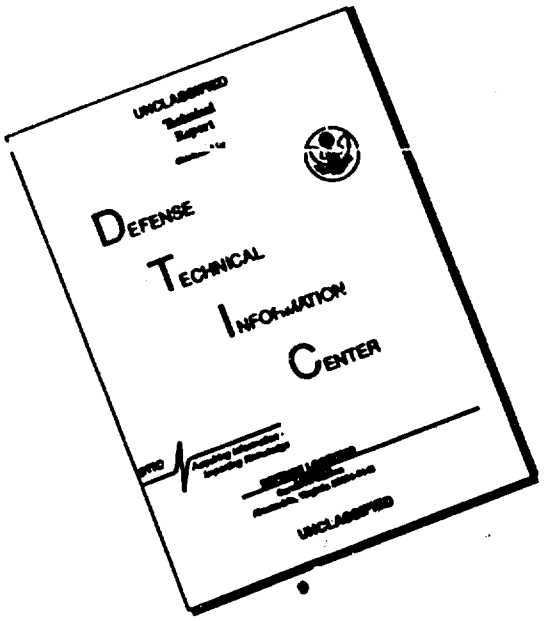
The Solvability of the Decision Problem for Classes of Proper Formulas and Related Results

R. A. DiPaola

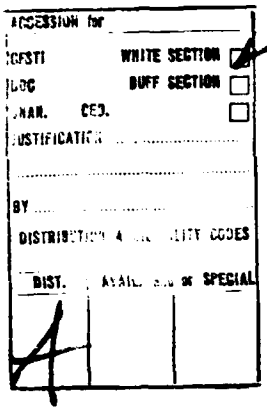
A Report prepared for

UNITED STATES AIR FORCE PROJECT RAND

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.



This research is supported by the United States Air Force under Project Rand—Contract No. F44620-67-C-0045—Monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of Rand or of the United States Air Force.

DOCUMENT CONTROL DATA

1. ORIGINATING ACTIVITY The Rand Corporation	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

THE SOLVABILITY OF THE DECISION PROBLEM FOR CLASSES OF PROPER FORMULAS AND RELATED RESULTS

4. AUTHOR(S) (Last name, first name, initial)

DiPaola, R. A.

5. REPORT DATE	6a. TOTAL NO. OF PAGES	6b. NO. OF REFS.
August 1971	43	17

7. CONTRACT OR GRANT NO.	8. ORIGINATOR'S REPORT NO.
F44620-67-C-0045	R-803-PR

9a. AVAILABILITY/LIMITATION NOTICES	9b. SPONSORING AGENCY
DDC-A	United States Air Force Project Rand

10. ABSTRACT

In connection with Rand's Relational Data File, a class of proper formulas has been proposed comprising those formalizations of questions to be processed by the RDF that are especially suitable for machine processing (~~discussed in R-511, RM-5428~~). That the decision problem for several classes of proper formulas is solvable was shown in ~~R-661~~. ~~This~~ report goes further and shows that the decision problem for the class lambda of proper formulas on the connectives negation, disjunction, conjunction, implication, and existential quantification is solvable. It follows that the decision problem for any class of proper formulas based on a subset of these connectives is solvable. Thus, for each of these classes, there is a mechanical decision procedure that determines whether an arbitrary formula is a member of the class. Also a representation theorem for the members of lambda is proven that reveals the reason for the existence of an algorithm that solves the decision problem for lambda.

11. KEY WORDS

Relational Data File
Linguistics
Information Retrieval
USSR--Cybernetics

R-803-PR
August 1971

The Solvability of the Decision Problem for Classes of Proper Formulas and Related Results

R. A. DiPaola

A Report prepared for

UNITED STATES AIR FORCE PROJECT RAND

Rand
SANTA MONICA, CA. 90406

PREFACE

A central part of Rand's Soviet Cybernetics Technology Project has been the development of the Relational Data File (RDF), "a computer-based system for the storage, retrieval, and logical analysis of factual data." Integral to the actual construction and implementation of the system was the theory originated by J. L. Kuhns [8, 9] to select the formulas of the predicate calculus that supposedly embody features facilitating machine processing. This theory governed the design of the programming language associated with the RDF. Accordingly, the classes of definite and proper formulas were defined. The definite formulas are characterized by a special semantic property and were proposed as the formalization of the "reasonable" inquiries to be processed by the RDF. Emphasis in the program was placed on the classes of proper formulas, each a subclass of the definite formulas, that also satisfy certain syntactic conditions judged to render them especially suitable for machine processing. The subclasses of proper formulas, in contradistinction to the class of definite formulas, depend on which logical primitives are employed. Different sets of primitives give rise to different classes of proper formulas. A formula is admissible if it is effectively transformable into a proper equivalent. Kuhns conjectured that every definite formula is admissible relative to a particular class of proper formulas. We have previously

shown [4] that the decision problem for the class of definite formulas is recursively unsolvable. Hence, no algorithm exists by which to determine whether an arbitrary formula is definite.

Also, in R-661 we showed that the decision problem for several classes of proper formulas is solvable. In this Report we go further and show that the decision problem for the class Γ of proper formulas on the connectives $\neg$, $\vee$, $\&$, $\supset$, $\exists$ is solvable. Hence, the decision problem for any class of proper formulas based on a subset of these connectives is solvable. Thus, there is a mechanical decision procedure which determines whether an arbitrary formula is a member of the class Γ. In addition, a representation theorem for the members of Γ is proven which lays bare the reason for the existence of an algorithm which solves the decision problem for Γ.

This study should be of particular interest to those concerned with the application of mathematical logic to data retrieval systems and to those responsible for the design and construction of retrieval systems intended for intelligence analysis.

SUMMARY

In this Report we show that the decision problem for the class Γ of proper formulas on the connectives $\neg, \vee, \&, \supset, ^*$ is solvable. It follows that the decision problem for any class of proper formulas based on a subset of these connectives is solvable. Thus, for each of these classes there is a mechanical decision procedure which determines whether an arbitrary formula is a member of the class. In addition, a representation theorem for the members of Γ is proven which lays bare the reason for the existence of an algorithm which solves the decision problem for Γ.

CONTENTS

THE SOLVABILITY OF THE DECISION PROBLEM FOR CLASSES OF PROPER FORMULAS AND RELATED RESULTS

1. INTRODUCTION

The Relational Data File (RDF) of The Rand Corporation is among the most developed of question-answering systems in terms of the size of the data base, the design or generation, and documentation of the associated files, and the specially constructed programming languages [6, 10, 11, 12, 13]. The "information language" of this system is an applied predicate calculus. The atomic units of information are binary relational sentences. The system has an inference-making capacity of the following sort: retrieval specifications embodying the rule of modus ponens are made by a user by means of the programming language Inferex. As an integral part of the design and implementation of this storage and retrieval system, a theory was developed by J. L. Kuhns to identify and systematically study the formulas of this calculus that represent the "reasonable" questions to put to a computer implementation of this system, with emphasis placed upon those representations that are supposedly especially suited for machine processing [8,9]. Accordingly, three classes of formulas were defined--definite, proper, and admissible. The classes of definite formulas are defined semantically and are invariant under the sentential and quantificational rules of the predicate calculus. Moreover, the notion of a

definite formula is essentially independent of the logical operators taken as primitives of the language, as is to be expected of a semantic notion. Indeed, languages $\mathfrak{L}_{C_1}$ and $\mathfrak{L}_{C_2}$ based on finite, functionally complete sets C_1 and C_2, respectively, of propositional and quantificational connectives satisfy the following property. There is an effective transformation of the formulas F of $\mathfrak{L}_{C_1}$ into formulas $\varphi(F)$ of $\mathfrak{L}_{C_2}$ such that the formula $F \equiv \varphi(F)$ is valid in all interpretations of the language $\mathfrak{L}_{C_1 \cup C_2}$. A fortiori, if Δ_{C_1} and Δ_{C_2} are the classes of definite formulas on C_1 and C_2, respectively, then $F \in \Delta_{C_1}$ iff $\varphi(F) \in \Delta_{C_2}$, and for each interpretation I of F and $\varphi(F)$, F and $\varphi(F)$ are satisfied in I at exactly the same instances. Therefore, we shall often write, "the class Δ of definite formulas."

As has been stated, the members of Δ share a semantic condition judged in [8] as being necessarily possessed by the symbolic representations of reasonable inquiries. To elaborate, the notion of a data base as defined in [8] amounts to the common notion of a <u>structure</u> with a finite number of relations; or, from the vantage point of a formal language, it is an interpretation of a finite number of predicate and constant symbols. The formulas F with free variables that are definite have the property that the sets of true instances of F in an interpretation I of F and in a special extension I' of I are the same. The formulas

without free variables are definite if their truth value is always preserved on passage from an interpretation I to an extension I' of I of the aforesaid special type. The precise definition is given in Sec. 2 below. In a paper which appeared in an earlier issue of this journal, we showed that the decision problem for the class Δ of definite formulas is recursively unsolvable [4].

The proper formulas are those definite formulas that satisfy a certain syntactic condition--namely, their principal subformulas must also be proper. Thus, all subformulas must be definite. The rationale behind this definition presumably runs as follows: The definite formulas mirror the reasonable inquiries, while the proper formulas are such that all their parts, i.e., subformulas, also have this desirable property. The admissible formulas are those that can be transformed into proper equivalents; that is, if F is admissible, there is a proper formula G such that for each finite interpretation I, with domain D, F and G are satisfied in I at exactly the same instances. Of course, it is clear a priori that the admissibility of a formula F is of little or no value unless the transformation φ of F into a proper equivalent G is effective.

The concept, then, of a proper formula involves the notion of a subformula. But determining which consecutive parts of a given formula are subformulas is, of course, a syntactic notion and depends on the identity of sentential

and quantificational connectives employed. Thus, the subclasses of proper formulas, in contradistinction to the class of definite formulas, depend on which of the logical connectives are taken as primitives. In this paper we show that the decision problem for various classes of proper formulas is solvable. We previously showed in [5] that the class of proper formulas on $\neg$, $\vee$, $\exists$, or $\neg$, $\supset$, $\exists$, or $\neg$, $\vee$, $\supset$,$\exists$ is recursive. However, the results of the present paper go beyond those of [5], allowing us to include conjunction among the connectives. Indeed, the proofs presented in [5] do not suffice to establish the results contained herein, while the latter results subsume those of [5].

In [8,9], stress is placed upon the particular class π_o of proper formulas on $\vee$, &, $\not\supset$ ("but not"), $\exists$. At this writing, the question of the decision problem for π_o remains open. We point out that for us the interest of these questions lies in this: They constitute a number of mathematical problems naturally arising in a practical context which have required the use of nontrivial theorems of mathematical logic for their resolution.

2. DEFINITIONS

The language $\mathfrak{L}$ that we use is the language of the full, pure first-order predicate calculus without equality, augmented with infinitely many individual constants. A formula in the predicate symbols $P_1^{n_1}, P_2^{n_2}, \ldots, P_t^{n_t}$, where the superscript denotes the rank or degree of the predicate symbol, and in the constants $c_1, c_2, \ldots, c_k$ is any formula F whose only symbols, other than sentential connectives, quantifiers, and individual variables, occur among $P_1^{n_1}, P_2^{n_2}, \ldots, P_t^{n_t}$, $c_1, \ldots, c_k$. An interpretation of F is a system $I = \langle D; R_1^{n_1}, \ldots, R_t^{n_t}; d_1, \ldots, d_k \rangle$, where D is a nonempty set; each relation $R_i^{n_i}$ is defined on D^{n_i} and assigned to $P_i^{n_i}$; and each d_j is assigned to c_j, $i = 1, \ldots, t$; $d_j \epsilon D$, $j = 1, \ldots, k$. An interpretation I of F is said to be finite if the domain D of I is finite. Developments of the notion of interpretation or structure may be found in [14] or [15].

2.1. Definition

If F is a formula with m free variables and $I = \langle D; R_1^{n_1}, R_2^{n_2}, \ldots, R_t^{n_t}; d_1, \ldots, d_k \rangle$ is a finite interpretation of F, T(F, I) is the set of members of D^m that satisfy F, if $m > 0$. If $m = 0$, we call F a <u>sentence</u>, and T(F, I) = t (truth) and T(F, I) = f (falsity) according to whether F is satisfied or not satisfied in I. We say a sentence F is <u>finitely satisfiable</u> if there is a finite interpretation in which it is satisfied; F is <u>finitely valid</u> if it is satisfied in all finite interpretations.

2.2 Definition

Let F be a formula and $I = \langle D; R_1^{n_1}, \ldots, R_t^{n_t}; d_1, \ldots, d_k \rangle$ a finite interpretation of F. Let * be an individual not in D. A *-extension I' of I for F is the interpretation $I' = \langle D'; S_1^{n_1}, \ldots, S_t^{n_t}; d_1, \ldots, d_k \rangle$, where $D' = D \cup \{*\}$ and $S_i^{n_i}$ is the extension of $R_i^{n_i}$ from D^{n_i} to D'^{n_i}, such that $S_i^{n_i}$ is false on any member of D'^{n_i} that has * among its components.

2.3. Definition

A formula F is said to be definite if, for all finite interpretations I of F, T(F, I) = T(F, I'), where I' is a *-extension of I for F. This definition is given in [8] in a different terminology.

Thus, the theorems and refutables of the predicate calculus that are sentences are definite. Clearly, all atomic formulas of $\mathfrak{L}$ are definite. Closure properties of the class of definite formulas are investigated at length in [8].

To describe the informal situation that this last definition is meant to reflect, we quote from [8]:

> Consider the question
> Who did not write Meaning and Necessity?
> We would certainly regard this as unreasonable, but how is the computer to know this? Should the computer simply list all the names in the dictionary (except 'R. Carnap') or should it somehow prohibit the question?
> The symbolic form of (1) is the formula
>
> NOT (Wxb) (2)

The difficulty with this expression is that its value set depends on more than the value set of the component 'Wxb'; it depends on other names in the dictionary. That is, if we add a new name to the dictionary the value set of (2) will change. A formula without this objectionable property will be called definite.

Other examples of indefinite formulas are easy to construct in terms of procedural requests. For example, the request

EXTRACT (x,y): x wrote Meaning and Necessity or y wrote Meaning and Necessity.

leads to the formula

(Wxb) or (Wyb) (4)

This is indefinite because any ordered pair of names gives a member of the value set providing only that 'a' is in the first or second position.

Let us now replace the rather vague notion of what constitutes a 'reasonable' request by the specific notion of definite formula. The problem is then to invent algorithms for identifying definite formulas and calculating their value sets [8], pp. 8-9.

. . . (The) data base is a set of elementary sentential formulas of degree zero. This distinguished set is called the file. The motivation is that the file shall consist of our stock of "true" sentences....

Value Sets of Elementary Sentential Formulas. We begin with the definition of value set. Since the generation of value sets will be a machine routine, we assign a special designation 'ω' to this operation. The first step is to define $\omega(\underline{f})$ for every $\underline{f}$ of degree zero.

What value should we assign to such a formula? We would like the value to correspond somehow to the "true value" . . . .

One method is to assume that the file gives an exhaustive description of the state of our universe of discourse. That is, the elementary formulas in the file tell us exactly which of the given individuals have the given properties, and which have the given relations between them. The value of an elementary formula is then "true" if the formula is in the file, and "false" otherwise [8], pp. 30-31.

Here the terms "data base," "dictionary," "file," "elementary sentential formulas of degree zero," and "value set of a formula" correspond, respectively, in our terminology to "interpretation," "constant symbols naming elements of an interpretation," "set of relations of an interpretation," "atomic formula without free variables (atomic sentence)," and "set of satisfying instances of a formula." The truth definition given in [8] of an elementary sentential formula A requires that any elementary sentential formula having the same predicate symbol as A be false if it contains a name not present in the file. This is reflected in the requirement of Definition 2.2 that the relation which interprets an atomic formula in a finite interpretation I be false on members of the relation which contain * in an *-extension I' of I. In [8] the truth definition of an arbitrary formula is built from that of elementary sentential formula by the rules of substitution, and the sentential and quantificational connectives in the usual way.

2.4. Remark

We recall again the following theorem [4]:

If A_k, $k \geq 0$, is the class of definite formulas on $\mathfrak{L}$ with k free variables, then A_k is not recursively enumerable.

Thus, the decision problem for each A_k is recursively unsolvable.

2.5. Examples

Consider the following formula F:

$\forall x \exists y P(x, y) \supset \exists y \forall x P(x, y)$. Let $D = \{1, 2\}$ and P be interpreted in D by the binary relation R true on the pairs (1, 2) and (2, 1), and false on the pairs (1, 1) and (2, 2). Thus, in this case $I = \langle D; R \rangle$. It is easily seen that $\forall x \exists y P(x, y)$ is true in I, whereas $\exists y \forall x P(x, y)$ is false in I. By the truth-table for implication, $\forall x \exists y P(x, y) \supset \exists x \forall x P(x, y)$ is consequently false in I.

We form the *-extension $I' = \langle D'; S \rangle$ of I by taking $D' = \{1, 2, *\}$ and S to be the binary relation which agrees wtih R on D x D and is false on the pairs (1, *), (*, 1), (2, *), (*, 2), and (*, *). Thus, $\forall x \exists y P(x, y)$ is false in I' since S(*, *), S(*, 1), and S(*, 2) are all false in D'. Similarly, $\exists y \forall x P(x, y)$ is false in I'. Hence F is true in I'. Since F is false in I but true in the *-extension I' of I we see that F is finitely satisfiable, is not finitely valid, and is not definite.

Consider the formula G:

$\exists x P(x) \ \& \ \forall x [P(x) \supset Q(x)]$. Let D be any nonempty finite set. Suppose $I = \langle D; R_1, R_2 \rangle$ is an interpretation in which R_1 and R_2 are unary relations serving as the interpretations of P and Q, respectively. If R_1 is false throughout D, then $\exists x P(x)$ and hence G are false in I. Thus, G is not finitely valid. But if, for example, R_1 holds for but a single member of D and R_2 is universally true in D, the G holds in I. Thus, G is finitely satisfiable.

Suppose that $I' = \langle D'; S_1, S_2 \rangle$ is a *-extension of I. Assume that G is false in I. If $\exists x P(x)$ is false in I, then since $S_1(*)$ is false, $\exists x P(x)$ and hence G are false in I'. If $\exists x P(x)$ is true in I but $\forall x(P(x) \supset Q(x))$ is false in I, then there is an element $d \in D$ such that S_1 is true on d and S_2 is false on d. Hence $S_1(d)$ is true and $S_2(d)$ is false in D'. Thus, $\forall x(P(x) \supset Q(x))$ is false in I'. Therefore, G is false in I'. Assume, now, that G is true in I. Then $\exists x P(x)$ is true in I and hence true in I'. By assumptions, $\forall x(P(x) \supset Q(x))$ is true in I. S_1 and S_2 are both false on *. So, $\forall x(P(x) \supset Q(x))$ is true in I'. Therefore G is true in I'. Since I is arbitrary, it follows that G is definite.

2.6. Definition

Let $u_1, u_2, \ldots, u_k$ be the unary connectives and $b_1, b_2, \ldots, b_\ell$ the binary connectives of the language $\mathfrak{L}$. It is assumed that at least one of $\exists$ (the existential quantifier) and $\forall$ (the universal quantifier) is among $u_1, u_2, \ldots, u_k$. We inductively define the property of being a subformula of a formula A of $\mathfrak{L}$.

(1) A is a subformula of A.

(2) If A is $u_i(B)$ and u_i is a propositional connective, then each subformula of B is a subformula of A, $i = 1, 2, \ldots, k$.

(3) If A is $u_i x(B)$, u_i is either $\exists$ or $\forall$, and x is a variable that occurs free in B, then each subformula of B is a subformula of A, $i = 1, 2, \ldots, k$.

(4) If A is $b_j(B, C)$, then each subformula of B is a subformula of A and each subformula of C is a subformula of A, $j = 1, 2, \ldots, \ell$.

(5) A formula B is a subformula of A only as prescribed by (1) through (4) above.

2.7. Notation

To facilitate the frequent use of certain terms, we shall write "fv," "fs," "nfv,""nfs," "re," for "finitely valid," "finitely satisfiable," "not finitely valid," "not finitely satisfiable," and "recursively enumerable," respectively. Also, we write "$F_1 \overset{I}{\Leftrightarrow} F_2$" to mean that, for each finite interpretation I, the formulas F_1 and F_2 hold at exactly the same instances of the domain of I. If $\supset$ is among our primitive connectives, we write "$A \equiv B$" for "$(A \supset B) \& (B \supset A)$." "$A \subset B$" means A is a proper subset of B.

2.8. Definition

A formula A of $\mathfrak{L}$ is proper on a set C of connectives iff each subformula of A is definite on C. This definition differs from the definition of "proper formula" given in [8, 9], but the two are easily seen to be equivalent.

Suppose C is a set of logical connectives which has $\vee$, $\&$, $\neg$, $\supset$, and $\exists$ among its members. For purposes of subsequent reference we state here a characterization theorem for proper formulas [9]:

If A and B are proper formulas on C, then

(1) $A \vee B$ is proper on C if (a) $Fv(A) = Fv(B)$, or (b) if

for one of A, B, say A, $Fv(A) \subset Fv(B)$ and A is nfs, or (c) both A and B are nfs, where "Fv(F)" denotes the set of free variables of the formula F.

(2) A & B is proper on C.

(3) $\neg$ A is proper on C iff A is a sentence.

(4) $A \supset B$ is proper on C iff (a) A and B are sentences, or (b) A is fv. (If A is fv, since A is proper, A is definite and fv, and hence a sentence.)

(5) $\exists x A$ is proper on C, where x is any variable.

This theorem is stated in [9] with all occurrences of "proper" replaced by "definite." As stated in [9], the present version follows at once from the definition of proper formula.

By (1) (a), (2), and (5) the sentence $\exists x \exists y((P(x) \& Q(x,y)) \vee R(x,y))$ is a proper formula in prenex form. The formula $\neg \exists y(\exists x P(x,y) \vee Q(y))$ is proper by (5), (1)(a), (5), and (3). The formula $\exists x \exists y((P(x) \& Q(x,y)) \supset Q(x,y)) \supset R(y)$ is proper by (4)(b), and (5). The formula $\neg P(x) \supset P(x)$ is not proper, though it is truth-functionally equivalent to the proper formula $P(x) \vee Q(x)$. In the following, we are mainly interested in proper formulas on the more familiar sets of connectives.

2.9. Definition

We define the depth d of a formula F.

(1) If F is atomic, $d(F) = 0$.

(2) If u is a unary primitive connective of the language $\mathcal{L}$, and F is u(A), then $d(F) = d(A)+1$.

(3) *If b is a binary primitive connective of £, and* F is b(A,B), then d(F) = d(A)+d(B)+1.

3. RESULTS

We have previously noted that the classes of proper formulas depend on which logical connectives are taken as primitives. Thus, classes π_1 and π_2 of proper formulas employing different but, in the usual sense, equivalent sets of connectives may have different properties. A result proven in [5] shows that this dependence is not absolute, and we cite it here for the sake of completeness.

THEOREM 3.1. The class π_p of proper formulas in prenex form defined on any complete set of connectives is recursive.

Before proceeding to the proof that various classes of proper formulas are recursive, we make some general observations which bear on the matter. Suppose that Δ is some nonrecursive class of formulas and that Γ is the subclass of Δ consisting of those formulas Δ all of whose subformulas are also members of Δ. We may ask whether Γ is recursive? Suppose further that $\bar{\Delta}$, the complement of Δ in the class of all formulas, is re. Thus, under an arithmetization of our formalism the predicate of natural numbers denoting membership in Δ is the negation of an re predicate. Hence, the natural predicate denoting membership in Γ is also the negation of an re predicate. Thus, one might expect the decision problem for Γ to also be

recursively unsolvable, that is, that Γ not be recursive and, hence, not re, since if $\bar{\Delta}$ is re, so is $\bar{\Gamma}$. Moreover, there are instances of classes of Δ and the derived class Γ, which are not re with $\bar{\Delta}$ and $\bar{\Gamma}$ both re. There are also instances in which both Δ and Γ are re but not recursive with $\bar{\Delta}$ and $\bar{\Gamma}$ both not re. We came upon these examples as a result of our attempts to prove that the decision problem for various classes of proper formulas is unsolvable. Indeed, these examples arise naturally as a result of consideration of the decision problem for any class of proper formulas. We therefore include them here for the sake of completeness, and because they are of some interest in their own right. Thus, let us take Δ to be the class of all formulas which are not valid. By the Godel completeness theorem, Δ consists of the formulas of the predicate calculus which are not theorems. By the fundamental result of Church and, independently, Turing, Δ is not re [2,17]. Γ is the class of all formulas none of whose subformulas is valid. With these definitions of Δ and Γ, we have

THEOREM 3.2. Γ is not re.

Proof. Let us recall the basic fact that a formula F is valid in a nonempty domain D if and only if the Skolem normal form of F is valid in D [3, pp. 224, 230]. We define Δ_s to be the class of those members G of Δ such that G is in prenex form and each existential quantifier of F precedes

every universal quantifier. (We do not say that Δ_s consists of those members of Δ in Skolem normal form because the Skolem normal form of a formula has no free variables [3]. Under our definition, Δ_s may include formulas with free variables and quantifier-free formulas. Of course, formulas in Skolem normal form are also in Δ_s.) Consider an arbitrary formula F. Let SK(F) be the Skolem normal form of F. By Godel's completeness theorem and the result cited above, F is a nontheorem if and only if SK(F) is a nontheorem; that is, $F \in \Delta \leftrightarrow SK(F) \in \Delta_s$. Therefore since Δ is not recursive, it follows that Δ_s is not recursive. The class Λ of formulas which are either not in prenex form, or, if in prenex form, have a prefix in which some universal quantifier precedes some existential quantifier, is recursive. The class $\bar{\Delta}$ of valid formulas (theorems) is re. Now, $\bar{\Delta}_s = \bar{\Delta} \cup \Lambda$. Thus $\bar{\Delta}_s$ is re. Hence, Δ_s is not re. We define Γ_s to be the class consisting of those members of Δ_s all of whose subformulas are also members of Δ_s. Of course, $\Gamma_s \subseteq \Delta_s$. But, each subformula of a member of Δ_s is itself a member of Δ_s. Hence, $\Gamma_s = \Delta_s$, and therefore Γ_s is not re. A recursive enumeration f of Γ_s is constructible from any recursive enumeration g of Γ by simply discarding those values of g which are in Λ. But Γ_s is not re. Hence, Γ is not re. Q.E.D.

COROLLARY 3.3. To decide of an arbitrary formula whether it has at least one valid subformula is recursively unsolvable.

By using the Skolem normal form for satisfiability [3, pp. 230-231], one can similarly prove

THEOREM 3.4. The Class Γ of formulas all subformulas of which are satisfiable is not re.

COROLLARY 3.5. To decide of an arbitrary formula whether it has at least one unsatisfiable subformula is recursively unsolvable.

There are examples of classes Δ and Γ with unsolvable decision problems in which Δ is re. Let us take Δ to the class of formulas which are nfv (We remind the reader of the notational conventions stated in 2.7). Γ is the class of those members of Δ all of whose subformulas are also members of Δ; thus Γ is the class of all formulas all subformulas of which are nfv.

With these definitions of Δ and Γ we have

THEOREM 3.6. Γ is not recursive.

Proof. We note in passing that Δ is re, and it follows easily that Γ is also re. We again define Δ_s as the class of all those members of Δ which are in prenex form such that each existential quantifier precedes every

universal quantifier. As in the proof of Theorem 3.2, if F is an arbitrary formula, $F \in \Delta \leftrightarrow SK(F) \in \Delta_s$, where "SK(F)" again designates the Skolem normal form of F. Γ_s is defined as in Theorem 3.2 as the class of all those members of Δ_s each of whose subformulas is in Δ_s. Again, as in Theorem 3.2, $\Delta_s = \Gamma_s$. By the well-known result of Trachtenbrot, Δ is not recursive [16]. Therefore, since again $F \in \Delta \leftrightarrow SK(F) \in \Gamma_s$, Γ_s is not recursive. Consequently, Γ is not recursive. For, suppose that Γ were recursive. Let Λ be defined as in the proof of Theorem 3.2. So Λ is recursive. To test if a formula $F \in \Gamma_s$, we ask first if F is in Λ. If $F \in \Lambda, F \notin \Gamma_s$. If $F \notin \Lambda$, then $F \in \Gamma_s$ if and only if $F \in \Gamma$. Thus, if Γ is recursive, then Γ_s is recursive. But Γ_s is not recursive. Hence Γ is not recursive. Q.E.D.

COROLLARY 3.7. The class Γ of all formulas which have at least one subformula which is fv is not re.

Similarly, using the Skolem normal form for satisfiability one can prove

THEOREM 3.8. The class Γ of formulas all subformulas of which are fs is not recursive.

COROLLARY 3.9. The class Γ of formulas which have at least one subformula which is nfs is not re.

We note that all of the above results are independent of which logical operators are taken as the primitives of the language $\mathfrak{L}$. Now, if C is any functionally complete set of connectives, let Δ_C be the class of definite formulas on C and Γ_C the class of members of Δ_C all subformulas of which are also members of Δ_C. Thus, Γ_C is the class of proper formulas on C. Then Δ_C is not re [4], and by the above considerations, one might not expect Γ_C to be re either. Nevertheless, taking $C = C_o = \{\neg, \vee, \&, \exists\}$, we shall prove that Γ_{C_o} is recursive.

DEFINITION 3.10. A formula F is said to be universal if F is in prenex form and if each quantifier occurring in the prefix of F is universal. A universal sentence is a universal formula with no free variables.

DEFINITION 3.11. A formula F is said to be existential if F is in prenex form and if each quantifier occurring in the prefix of F is existential. An existential sentence is an existential formula with no free variables. Universal (existential) formulas are said to be nonvacuous if they contain at least one universal (existential) quantifier.

DEFINITION 3.12. The class $\mathcal{D}$ of formulas (on $\neg, \vee, \&, \exists, \forall$) consists of all formulas F such that

(1) All occurrences of quantifiers are in either universal or existential subformulas of F.

(2) Each occurrence of a negation sign occurs in subformulas of F which are nonvacuous universal or existential sentences.

(3) No free variable of F occurs within the scope of a universal quantifier.

(4) Each bound variable of F is bound by only a single occurrence of a quantifier and no variable in F occurs both free and bound.

We note that it follows from (1) that each subformula of F in prenex form is either universal or existential. Thus a typical F in $\mathcal{B}$ consists of universal sentences, existential sentences, and existential formulas with free variables joined together by occurrences of & or V.

EXAMPLES

The formula

$$[\exists x\exists y((P(x)\&Q(y))\vee R(y,z))\&\forall u\forall v\forall w(\neg((P(u)\vee S(u,v))\&R(w,v)))]$$
$$\vee\exists r\exists s((R(r,s)\&P(s))\vee Q(r))$$

is a member of $\mathcal{B}$, whereas the formula

$$[(\exists x\exists y(P(x)\&Q(y))\vee\exists yR(y,z))\&\forall u\forall v\forall w(\neg((P(u)\vee S(u,v))\&R(w,v)))]$$
$$\vee\exists r\exists s((R(r,s)\&P(s))\vee Q(r))$$

is not, since condition (4) is violated. Nor is the formula

$$[\exists x \exists y((P(x) \& Q(y)) \vee R(y,z)) \& \forall u \forall v \forall w (\neg((P(u) \vee S(u,v)) \& R(w,r)))]$$
$$\vee \exists v \exists w((R(v,w) \& P(w)) \vee Q(v)),$$

since again (4) is violated.

The formula $\forall x(\neg P(x) \& (\exists y R(x,y) \vee Q(z)))$ fails to be in $\mathcal{B}$ on three counts; conditions (1), (2), and (3) are violated. The formula

$$\forall x \neg P(x,x) \& \exists z P(t,z) \& \forall u \forall v \forall w((\neg P(u,v) \vee \neg P(v,w)) \vee P(u,w))$$

is a member of $\mathcal{B}$.

We recall that "Γ_{C_o}" designates the class of proper formulas on $C_o = \{\neg, \vee, \&, \exists\}$. We now establish a representation theorem for Γ_{C_o}, which we state as

LEMMA 3.13. Each member of Γ_{C_o} is provably equivalent to some member of $\mathcal{B}$ (in some standard axiomatization of the predicate calculus on $\mathcal{L}$); equivalently, for each formula $F \in \Gamma_{C_o}$, there is a formula $G \in \mathcal{B}$ such that the formula $F \equiv G$ is valid (See 2.7). Also F and G have the same free variables.

Proof. Consider a formula $F \in \Gamma_{C_o}$. The proof proceeds by induction on the depth d of F. Suppose $d = 0$. Then F is atomic. Conditions (1) to (4) defining membership in $\mathcal{B}$ are clearly satisfied by atomic formulas, so $F \in \mathcal{B}$ if $d = 0$. Since $F \equiv F$ is of course valid, the lemma holds for $d = 0$.

Suppose that the lemma holds for all members of Γ_{C_o} of depth $d' < d$, $d > 0$. Consider a formula F in Γ_{C_o} of depth d.

Case 1. F is $A \vee B$. Then $d(A) < d$ and $d(B) < d$. By the induction hypothesis, there are formulas A' and B' in $\mathcal{B}$ such that the formulas $A \equiv A'$ and $B \equiv B'$ are valid. Also, $Fv(A) = Fv(A')$ and $Fv(B) = Fv(B')$, where "Fv(F)" denotes the set of variables occurring free in a formula F. It is immediate that $(A \vee B) \equiv (A' \vee B')$ is valid. Consider a formula B" obtained from B' by a change of bound variables such that (1) every bound variable of B" differs from every bound variable of A' and also from every free variable of A' and (2) so that condition (4) of 3.12 remains satisfied. B' and B" differ in at most their bound variables, so that $B' \equiv B"$ is valid. (In terms of [7] p. 153, B" is congruent to B'.) Also, $B" \in \mathcal{B}$. The formulas $A \vee B$ and $A' \vee B"$ have the same free variables and $(A \vee B) \equiv (A' \vee B")$ is valid, since $(A' \vee B") \equiv (A' \vee B')$ is valid. It is not difficult to see that $A' \vee B"$ is a member of $\mathcal{B}$. Also $Fv(F) = Fv(A' \vee B")$.

Case 2. F is $A \& B$. To treat this case we proceed just as in Case 1 and by induction hypothesis obtain formulas $A' \in \mathcal{B}$ and $B' \in \mathcal{B}$ such that $A \equiv A'$ and $B \equiv B'$ are valid formulas and $Fv(A) = Fv(A')$ and $Fv(B) = Fv(B')$. Again, a formula is defined as in Case 1 so that $A \& B$ and $A' \& B"$ have the same free variables and $(A \& B) \equiv (A' \& B")$ is valid. It is easy to see that $(A' \& B") \in \mathcal{B}$.

Case 3. F is $\exists x A$. By the induction hypothesis, there is a formula A' in $\mathcal{B}$ such that the formula $A \equiv A'$ is valid and $Fv(A) = Fv(A')$. We may assume that x occurs free in A. Since $A' \in \mathcal{B}$ all bound variables of A' differ from x. Suppose that $S_1, S_2, \ldots, S_n$ are all the nonvacuous universal or existential sentences that occur as subformulas of A'. Suppose that $F_1, F_2, \ldots, F_m$ are all the nonvacuous existential formulas with free variables that occur as subformulas in A' such that no F_j, $j = 1, 2, \ldots, m$, occurs as a subformula of any existential subformula of A' other than itself. Thus, the F_j are the "biggest" existential subformulas of A' that are not sentences and not subformulas of existential sentences. No S_i is a subformula of any F_j, and, from the definition of the F_j, no F_j is a subformula of any S_i, $i = 1, 2, \ldots, n$ $j = 1, 2, \ldots, m$. We note that all occurrences of negation signs in A' are internal to some S_i, $i = 1, 2, \ldots, n$. By the definition of $\mathcal{B}$, the remaining subformulas of A' are quantifier-free, that is, those which are subformulas of no S_i or F_j, $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$.

Consider $m + n$ proposition letters $P_1, P_2, \ldots, P_n$, $Q_1, Q_2, \ldots, Q_m$, none of which occurs in F. Let A" be the formula which results from A' by replacing each occurrence of S_i in A' by P_i and each occurrence of F_j in A' by Q_j, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$. Let B be a disjunctive normal form of A" and A"' the formula which results by replacing each occurrence of P_i in B by S_i and

each occurrence of Q_j in B by F_j, i = 1, 2, ..., n
j = 1, 2, ..., m. Thus, A'" amounts to putting A' into disjunctive normal form, treating the S_i and F_j as atoms. A'" has the form $D_1 \vee D_2 \vee \ldots \vee D_k$; each D_a is of the form $C_1 \& C_2 \& \ldots \& C_p$, where each C_b is either an atom which occurs in A' outside of any S_i or F_j (Such an atom may occur in some S_i or F_j as well, but this is irrelevant), or C_b is some F_j, or C_b is some S_i, b = 1, 2, ..., P. Without loss of generality we may assume that for some q $1 \leq q \leq p$, each of C_q, C_{q+1}, ..., C_p is an atom not containing x, an F_j not containing x free, or an S_i; and that all atoms not containing x, all F_j not containing x free, and all the S_i that occur in D_a occur among C_q, C_{q+1} &...& C_p, The formulas $\exists x A \equiv \exists x A'$ and $\exists x A' \equiv \exists x A'''$ are valid. Also, $\exists x A''' \equiv (\exists x D_1 \vee \exists x D_2 \vee \ldots \vee \exists x D_k)$ is valid. Now, $\exists x D_a$ is $\exists x(C_1 \& C_2 \& \ldots \& C_q \& C_{q+1} \& \ldots \& C_p)$. By the above, the formula $\exists x D_a \equiv \exists x(C_1 \& C_2 \& \ldots \& C_{q-1}) \& C_q \& C_{q+1} \& \ldots \& C_p$ is valid. Using the fact that A' ϵ $\mathcal{D}$ and hence that each bound variable of $C_1 \& C_2 \& \ldots \& C_{q-1}$ is bound by a single quantifier, we bring all existential quantifiers x_1, $\exists x_2$, ..., $\exists x_r$ which occur in $C_1 \& C_2 \& \ldots \& C_{q-1}$ to the front and arrive at the formula

$$E_a: \exists x \exists x_1 \ldots \exists x_r (B_1 \& B_2 \& \ldots \& B_{q-1}) \& C_q \& C_{q+1} \& \ldots \& C_p,$$

and note that $\exists x D_a \equiv E_a$ is valid. Also, it is easy to check that E_a is in $\mathcal{D}$. This analysis applies to each formula $\exists x D_a$, a = 1, 2, ..., k. Thus, we arrive at

E_1, E_2, ..., E_k, all in $\mathcal{B}$, such that $(\exists xD_1 \vee \exists xD_2 \vee \ldots \vee \exists xD_k) \equiv (E_1 \vee E_2 \vee \ldots \vee E_k)$ is valid. Now, we perform in the obvious way a change of bound variables on each E_a in consecutive order, $a = 1, 2, \ldots, k$, arriving at formulas E_1', E_2', ..., E_k' so that (1) each E_a' is congruent to E_a and hence $E_a \equiv E_a'$ is valid and $E_a' \in \mathcal{B}$, $a = 1, 2, \ldots, k$; (2) no bound variable occurs in both members of any pair E_a', E_b', $1 \leq a,b \leq k$, $a \neq b$; and (3) no variable occurs free and bound in $E_1' \vee E_2' \vee \ldots \vee E_k'$. It follows that $E_1' \vee E_2' \vee \ldots \vee E_k'$ is a member of $\mathcal{B}$. We designate $E_1' \vee E_2' \vee \ldots \vee E_k'$ as F_k. From the above, it follows that $\exists xA' \equiv F_k$ is valid, and hence $\exists xA \equiv F_k$ is valid. Also, since throughout the argument no free variables have been deleted or introduced, $Fv(F) = Fv(F_k)$.

Case 4. F is $\neg A$. Since $F \in \Gamma_{C_o}$, by (3) of 2.8 we see that A is a sentence. By the induction hypothesis, there is a sentence A' in $\mathcal{B}$ such that the formula $A \equiv A'$ is valid. By repeated use of the standard facts that for any formulas B, C, the formulas $\neg \exists xB \equiv \forall x \neg B$, $\neg \forall xB \equiv \exists x \neg B$, $\neg(B \& C) \equiv (\neg B \vee \neg C)$, $\neg(B \vee C) \equiv (\neg B \& \neg C)$ are valid, we move the left-most negation sign in $\neg A'$ inward until we arrive at a sentence G in which all negation signs occur in quantifier-free subformulas of G, that is, no occurrence of $\neg$ in G has a quantifier within its scope. By these transformations each existential sentence of A' is transformed into a universal sentence and each universal sentence into an existential sentence. It is not hard to see that $\neg A' \equiv G$ is

valid. Hence, $F \equiv G$ is valid and, $Fv(F) = Fv(G)$, and $G \in \mathcal{B}$. Q.E.D.

LEMMA 3.14. If $F \in \Gamma_{C_o}$, then F has a prenex form in which every universal quantifier precedes every existential quantifier, and a prenex form in which every existential quantifier precedes every universal quantifier.

Proof. By the preceding lemma, there is a formula G in $\mathcal{B}$ such that the formula $F \equiv G$ is valid, where F and G have the same free variables. In G no existential quantifier occurs within the scope of any universal quantifier, and no universal quantifier occurs within the scope of an existential quantifier. Also, no quantifier falls within the scope of a negation sign. These conditions follow from the fact that G is a member of $\mathcal{B}$. Consequently, in performing prenex operations on G we are able to first pull out all the existential quantifiers and then pull out all the universal quantifiers. (There is no difficulty in pulling an existential quantifier over an occurrence of &, or a universal quantifier over an occurrence of $\vee$, since in G no bound variable occurs in more than one quantifier.) Consequently, G has a prenex form $\exists x_1 \exists x_2 \ldots \exists x_n \forall y_1 \forall y_2 \ldots \forall y_m M$ where M is quantifier free. Likewise, all universal quantifiers can be pulled out before all existential quantifiers are pulled out, resulting in a prenex form

$$\forall y_1 \forall y_2 \ldots \forall y_m \exists x_1 \exists x_2 \ldots \exists x_n M, \quad M \text{ quantifier free.}$$

But, since $F \equiv G$ is valid, any prenex form of G is a prenex form of F. Q.E.D.

LEMMA 3.15. If $F \in \Gamma_{C_o}$, it is decidable whether F is fv or F is fs (See 2.7).

Proof. By the preceding lemma, F has a prenex form of the type $\forall y_1 \forall y_2 \ldots \forall y_m \exists x_1 \exists x_2 \ldots \exists x_n M$, M quantifier free. By [1], pp. 70-71, if a formula F has a prenex form with a prefix of this type, then F is universally valid if it is valid in domains with 1, 2, 3, ..., m elements. Thus, F is finitely valid iff F is universally valid. This is decided by testing successively whether F is valid in domains with 1, 2, ..., m elements. Hence it is decidable whether F is fv.

A formula G is fs iff $\neg G$ is not fv. F has a prenex form $\exists x_1 \exists x_2 \ldots \exists x_n \forall y_1 \forall y_2 \ldots \forall y_m M$, M quantifier free. Hence, $\neg F$ has a prenex form $\forall x_1 \forall x_2 \ldots \forall x_n \exists y_1 \exists y_2 \ldots \exists y_m \neg M$. But as noted in the preceding paragraph it is decidable whether formulas having a prenex form of this type are fv; in fact, by testing for validity in domains with 1, 2, ..., n elements. Hence, it is decidable whether $\neg F$ is fv. Therefore, it is decidable whether F is fs. Q.E.D.

We define the proposition P(d): for each formula F on C_o of depth d, if all subformulas F of depth $d' < d$ are proper on C_o, then it is decidable whether F is proper on C_o.

LEMMA 3.16. P(d) holds for each d.

Proof. If $d = 0$, then F is atomic and proper on any set of connectives. Hence, P(0) holds. Consider an arbitrary formula F of depth d, $d > 0$. Assume the hypothesis of the

assertion; that is, that any subformula of F of depth $d' < d$ is proper on C_o. We recall that for any formula G, we designate the set of free variables in G by "Fv(G)."

Case 1. F is $A \vee B$. Since $d(A) < d$, $A \in \Gamma_{C_o}$ and $B \in \Gamma_{C_o}$. If $Fv(A) = Fv(B)$, then by (1)(a) of 2.8, $F \in \Gamma_{C_o}$. If $Fv(A) \subset Fv(B)$, then $F \in \Gamma_{C_o}$ iff A is nfs by (1)(b) of 2.8. By Lemma 3.15, it is decidable whether A is fs. Thus, it is decidable whether A is nfs. Hence, it is decidable whether $F \in \Gamma_{C_o}$. Similarly, if $Fv(B) \subset Fv(A)$. Suppose $Fv(A) \neq Fv(b)$, $Fv(A) \not\subset Fv(B)$, $Fv(B) \not\subset Fv(A)$. Then $F \in \Gamma_{C_o}$ iff A is nfs and B is nfs by (1)(c) of 2.8. By Lemma 3.15 it is decidable whether A is fs and B is fs. Hence, it is decidable whether A is nfs and B is nfs. We conclude it is decidable whether $F \in \Gamma_{C_o}$.

Case 2. F is $A \& B$. Again by our assumption, since $d(A) \subset d$, $d(B) < d$, we have $A \in \Gamma_{C_o}$ and $B \in \Gamma_{C_o}$. But then it follows by (2) of 2.8 that $F \in \Gamma_{C_o}$.

Case 3. F is $\neg A$. Then $d(A) < d$ and by our assumption $A \in \Gamma_{C_o}$. Hence, by (3) of 2.8 $F \in \Gamma_{C_o}$ iff $Fv(A) = \emptyset$.

Case 4. F is $\exists x A$. Then $d(A) < d$ and $A \in \Gamma_{C_o}$. By (5) of 2.8 we conclude $F \in \Gamma_{C_o}$. Q.E.D.

We finally obtain

THEOREM 3.17. Γ_{C_o} is recursive.

Proof. Consider an arbitrary formula F. Suppose that the depth of F is d: $d(A) = d$. By Lemma 3.16 we can successively determine whether all subformulas of F of depth $d' \leq d$ are proper on C_o. Hence, we can decide whether F is proper on C_o, that is, whether $F \in \Gamma_{C_o}$.

We now consider the decision problem for the class Γ_{C_1} of proper formulas on $C_1 = \{\neg, \vee, \&, \supset, \exists\}$. First we define a translation φ of the class of formulas on C_1 into the class of formulas on C_o.

$\varphi(A) = A$ if A is atomic.

$\varphi(A \vee B) = \varphi(A) \vee \varphi(B)$.

$\varphi(A \& B) = \varphi(A) \& \varphi(B)$.

$\varphi(\neg A) = \neg \varphi(A)$.

$\varphi(A \supset B) = \varphi(\neg A) \vee \varphi(B)$.

$\varphi(\exists x A) = \exists x \varphi(A)$.

By induction on the depth d of a formula F on C_1, we see that φ is defined for every formula on C_1.

LEMMA 3.18. For each formula F, the formula $F \equiv \varphi(F)$ is valid.

Proof. This is obvious. A detailed proof proceeds by induction on the depth d of F, employing a case analysis of just the sort given in the proofs of preceding lemmas. Q.E.D.

LEMMA 3.19. A is proper on C_1 iff $\varphi(A)$ is proper on C_o. In other words, $A \in \Gamma_{C_1}$ iff $\varphi(A) \in \Gamma_{C_o}$.

Proof. The proof proceeds by induction on the depth d of A. If d = 0, then A is atomic and $\varphi(A) = A$; hence, the assertion holds. Suppose the statement is true for all formulas B of depth $d' < d$, $d > 0$. Consider a formula A of depth d.

Case 1. A is $\neg B$. Then $\varphi(A)$ is $\neg \varphi(B)$. Assume A is proper on C_1; then B is proper on C_1. Then, by the induction hypothesis, $\varphi(B)$ is proper on C_o. Since $\neg B$ is proper, A is a sentence by (3) of 2.8. Hence, $\varphi(B)$ is a sentence and $\varphi(A)$ is proper on C_o. Likewise, if $\varphi(A) = \neg\varphi(B)$ is proper on C_o, $\varphi(B)$ is proper on C_o and a sentence. By definition of φ, B is a sentence; by the induction hypothesis, B is proper on C_1. Hence, A is proper on C_1.

Case 2. A is $\exists x B$. Then $\varphi(A) = \exists x \varphi(B)$. A and $\varphi(A)$ are proper on C_1, C_o, respectively, iff B and $\varphi(B)$ are proper on C_1, C_o, respectively. By the induction hypothesis, B is proper on C_1 iff $\varphi(B)$ is proper on C_0. Hence, A is proper on C_1 iff $\varphi(A)$ is proper on C_o.

Case 3. A is $B \supset C$. Then $\varphi(A)$ is $\neg \varphi(B) \vee \varphi(C)$.

Subcase 1. B and C are both sentences. Then $\varphi(B)$ and $\varphi(C)$ are sentences. A is proper on C_1 iff both B and C are proper on C_1 by (4)(a) of 2.8. By the induction hypothesis, B and C are proper on C_1 iff $\varphi(B)$ and $\varphi(C)$ are

proper on C_o. Since B is a sentence, B is proper on C_1 iff $\neg\varphi(B)$ is proper on C_o. But $\neg\varphi(B)\vee\varphi(C)$ is proper on C_o iff $\neg\varphi(B)$ and $\varphi(C)$ are proper on C_o by (1)(a) of 2.8. Therefore A is proper on C_1 iff $\neg\varphi(B)\vee\varphi(C)$ is proper on C_o.

Subcase 2. At least one of B, C is not a sentence. In this subcase, A is proper on C_1 iff B is a proper fv sentence on C_1 and C is proper on C_1 by (4)(b) of 2.8. Thus, if A is proper on C_1, C is not a sentence. Hence, $\varphi(C)$ is not a sentence and, by the induction hypothesis, $\varphi(C)$ is proper on C_o. Also, if B is a proper, fv sentence on C_1, then $\neg\varphi(B)$ is a nfs proper sentence on C_o by Lemma 3.18 and (3) of 2.8. Hence, $\neg\varphi(B)\vee\varphi(C)$ is proper on C_o by (1)(b) of 2.8. Similarly, since one of B, C is not a sentence, if $\neg\varphi(B)\vee\varphi(C)$ is proper on C_o, then $\neg\varphi(B)$ is a proper sentence on C_o by (3) of 2.8. Thus, C and $\varphi(C)$ have free variables, and hence at least $\neg\varphi(B)$ is nfs by (1)(b) of 2.8. Hence, B is, by the induction hypothesis, a proper sentence on C_1 and C is a proper formula on C_1 with free variables. Since $F \equiv \varphi(F)$ is valid in the predicate calculus for each formula F by Lemma 3.18, B is fv. Hence, A is proper on C_1 by (4)(b) of 2.8. We conclude that A is proper on C_1 iff $\neg\varphi(B)\vee\varphi(C)$ is proper on C_o. Q.E.D.

Case 4. A is B & C. By (2) of 2.8 A is proper on C_1 iff B is proper on C_1 and C is proper on C_1. By the induction hypothesis, B is proper on C_1 and C is proper

on C_1 iff $\varphi(B)$ is proper on C_o and $\varphi(C)$ is proper on C_o, respectively. Consequently, A is proper on C_1 iff $\varphi(A)$ is proper on C_o. Q.E.D.

We thus obtain

THEOREM 3.20. Γ_{C_1} is recursive.

Proof. Indeed, an arithmetization of our formalism, together with Lemma 3.19, shows that the decision problem for the class of proper formulas on C_1 is 1-1 reducible to the decision problem for the class of proper formulas on C_o. By Lemma 3.17, the latter class, that is, Γ_{C_o}, is recursive. Hence, Γ_{C_1} is recursive. Q.E.D.

We note that by Lemma 3.13 and Lemma 3.18, the assertion of Lemma 3.13 holds also for Γ_{C_1}; that is, if F is any member of Γ_{C_1} there can be effectively found a member $\varphi(F)$ of the class $\mathcal{B}$ such that the formula $F \sim \varphi(F)$ is valid.

4. REMARKS

In [8] p. vi it is stated that

> Among the definite formulas in a class of particular interest--the proper formulas. These formulas, because of their structure, are especially suitable for machine processing.... Finally, the important class of definite but improper formulas is studied. The approach is to transform these into proper equivalents. Those formulas for which this transformation can be done are called admissible. It is conjectured that every definite formula is admissible....

Here, by "proper formulas," the author refers to the class of proper formulas on $C = \{\vee, \&, \not\supset$ ("but not"), $\exists\}$. Part of Chapter 6 of [8] is devoted to defining admissibility transformations for various subclasses of definite formulas. By Theorem 3.20, it follows that if Δ is any class of definite formulas on any functionally complete set of connectives, then the analogue of this conjecture, as applied to Δ and the class Γ_{C_1}, is false in the following sense. There is no effective transformation φ such that $F \in \Delta \leftrightarrow \varphi(F) \in \Gamma_{C_1}$. The existence of such a φ would imply that Δ is recursive, which is not the case. In fact, for the same reason, Δ is not even Turing reducible to Γ_{C_1}. Since Γ_{C_1} is recursive, if C is any set of connectives such that $C \subset C_1$, then Γ_C is recursive. Δ is therefore not Turing reducible to any such class Γ_C.

All of the results of [5] on solvable classes of proper formulas are subsumed under Theorem 3.20. The principal advance of the present paper over [5] is embodied

in Lemmas 3.13, 3.14, and 3.15. These results correspond to Lemma 3.2 of [5] which establishes that if F is a proper formula on C = $\{\neg, \vee, \exists\}$, then it is decidable whether F is fv or fs. However, if & is included among the connectives the proof of 3.2 of [5] does not suffice to establish that it is decidable whether F is fv or fs. This is accomplished by Lemmas 3.13, 3.14, 3.15.

REFERENCES

1. Ackermann, W., "Solvable Cases of the Decision Problem," in L. E. J. Brouwer, E. W. Beth, A. Heyting (eds.) Studies in Logic and the Foundations of Mathematics, North-Holland Publishing Co., Amsterdam, 1954.

2. Church, Alonzo, "A Note on the Entscheidungsproblem," The Journal of Symbolic Logic, Vol. 1, No. 1 (1936), pp. 40-41.

3. _____, "Introduction to Mathematical Logic," Vol. I, Princeton University Press, Princeton, New Jersey, 1956.

4. Di Paola, R. A., The Recursive Unsolvability of the Decision Problem for the Class of Definite Formulas, The Rand Corporation, RM-5639-PR, April 1968 (Also published in JACM, Vol. 16, No. 2, April 1969, pp. 324-327).

5. _____, "The Relational Data File and the Decision Problem for Classes of Proper Formulas," Proc. of the Symposium on Information Storage and Retrieval, Ed. by Jack Minker, University of Maryland and Sam Rosenfeld, NASA, April 1-2, 1971, pp. 95-104.

6. Holland, Wade, Relational Data File: Input Techniques, The Rand Corporation, RM-5621-PR, February 1969.

7. Kleene, S. C., Introduction to Metamathematics, D. Van Nostrand Co., Princeton, New Jersey, 1952.

8. Kuhns, J. L., Answering Questions by Computer: A Logical Study, The Rand Corporation, RM-5428-PR, December 1967.

9. _____, Interrogating a Relational Data File: Remarks on the Admissibility of Input Queries, The Rand Corporation, R-511-PR, November 1970.

10. Levien, R. E., Relational Data File: Experience with a System for Propositional Data Storage and Inference Execution, The Rand Corporation, RM-5947-PR, April 1969.

11. _____, "Relational Data File II: Implementation," in G. Schector (ed.), Information Retrieval: A Critical View, Thompson Book Co., Washington, D. C., 1966.

12. _____, and M. E. Maron, "A Computer System for Inference Execution and Data Retrieval," Comm. ACM, Vol. 10, No. 11, November 1967, pp. 715-721.

13. Maron, M. E., "Relational Data File I: Design Philosophy," in G. Schector (ed.), Information Retrieval: A Critical View, Thompson Book Co., Washington, D. C., 1966.

14. Mendelson, E., Introduction to Mathematical Logic, D. Van Nostrand Co., New York, 1964.

15. Shoenfield, J. R., Mathematical Logic, Addison Wesley, Reading, Mass., 1967.

16. Trahtenbrot, V. A., "Impossibility of an Algorithm for the Decision Problem in Finite Classes," Amer. Math. Soc. Translations, Series 2, Vol. 23, 1963, pp. 1-5.

17. Turing, A. M., "On Computable Numbers, with an Application to the Entscheidungsproblem," Proc. of London Math. Soc., ser. 2, Vol. 42 (1936-7), pp. 236-265; corrections Ibid., Vol. 43 (1937), pp. 544-546.