7

# The Generalized Resolution Principle

J. A. Robinson
College of Liberal Arts
Syracuse University, N.Y.

## 1. INTRODUCTION

The generalized resolution principle is a single inference principle which provides, by itself, a complete formulation of the quantifier-free first-order predicate calculus with equality. It is a natural generalization of the various versions and extensions of the resolution principle, each of which it includes as special cases; but in addition it supplies all of the inferential machinery which is needed in order to be able to treat the intended interpretation of the equality symbol as 'built in', and obviates the need to include special axioms of equality in the formulation of every theorem-proving problem which makes use of that notion.

The completeness theory of the generalized resolution principle exploits the very intuitive and natural idea of attempting to construct counterexamples to the theorems for which proofs are wanted, and makes this the central concept. It is shown how a proof of a theorem is generated automatically by the breakdown of a sustained attempt to construct a counterexample for it. The kind of proof one gets depends entirely on the way in which the attempt to construct a counterexample is organized, and the theory places virtually no restrictions on how this shall be done. Consequently there is a very wide freedom in the form which proofs may take: the individual inferences in a proof may be very 'small' or very 'large' (in a scale of measurement which, roughly speaking, weighs the amount of computing necessary to check that the inference is correct). It is even correct to infer the truth of a true proposition *in just one step*, but, presumably, to offer such a proof to someone who wishes to be convinced of the proposition's truth would not be helpful epistemologically. His conviction would come, not from contemplating the proof itself, but rather from examining the computation which shows the correctness of its single inference step.

## 2. QUANTIFIER-FREE FIRST-ORDER PREDICATE CALCULUS WITH EQUALITY

### 2.1. Syntax

The expressions of the calculus are either simple or composite, and if they are composite they have a unique *applicative structure* consisting of two parts, an *operator* and an *operand.* The intention is that, when it is *interpreted* as explained in 2.2 below, every expression shall *denote* something, and that the entity denoted by a composite expression $AB$ (where $A$ is the operator, and $B$ the operand, of $AB$) shall always be the result of *applying* the entity denoted by $A$ *to* the entity denoted by $B$. The expressions are all built up from primitive symbols in a systematic way explained below.

**2.1.1.** *Vocabularies.* A *vocabulary* is a set $V$ of symbols, partitioned into disjoint subsets as follows: $I(V)$ is the set of *individual symbols* in $V$; and, for each natural number $n \geqslant 0$, the set $F_n(V)$ is the set of *function symbols of degree n* in $V$, and $R_n(V)$ is the set of *relation symbols of degree n* in $V$. It is possible that some, and even all but finitely many, of the sets $I(V)$, $F_0(V)$, $R_0(V)$, $F_1(V)$, $R_1(V)$, . . ., should be empty; but we assume that at least one of them is not. We shall usually employ lower case letters for individual symbols and upper case letters for function and relation symbols.

**2.1.2.** *Expressions over a vocabulary.* Let $V$ be a vocabulary. Then the *expressions over V* are the terms, sentences, sequences of terms, sets of terms and sets of sentences defined below, together with the members of $V$ themselves. In these definitions, the references to individual, function and relation symbols are to be taken as restricted to $V$.

**2.1.2.1.** *Terms.* A term is either an individual symbol or else has the form $FT$ where $F$ is a function symbol of degree $n$ and $T$ is a sequence of $n$ (not necessarily distinct) terms.

**2.1.2.1.1.** *Sequences of terms.* A sequence of $n$ terms is simply the empty string when $n=0$. When $n>0$, a sequence of $n$ terms is a parenthesized list $(T_1, \ldots, T_n)$ each component in which is a term. It is not necessary that all of the components be distinct.

**2.1.2.2.** *Atoms.* An atom either has the form $RT$ where $R$ is a relation symbol of degree $n$ and $T$ is a sequence of $n$ terms, or else is an *equation* $=S$ where $=$ is the equation symbol and $S$ is a nonempty set of terms.

**2.1.2.2.1.** *Sets of terms.* A set of terms is a list of terms enclosed in a pair of braces: $\{T_1, \ldots, T_n\}$. When no terms at all appear between the braces the set is said to be *empty.* A term is said to be *in* a set if and only if it is one of the components of the list, and two sets are regarded as being the same if every term which is in one of them is also in the other. In other words, the order and multiplicity of terms in a set is irrelevant.

**2.1.2.3.** *Literals.* A literal is either an atom or has the form $\neg A$ where $\neg$ is the negation symbol and $A$ is an atom. The literals $A$, $\neg A$ are *complementary*, and each is *the complement of* the other.

**2.1.2.4.** *Sentences.* A sentence is either an atom, or a conjunction ■$C$ where

■ is the conjunction symbol and $C$ is a set of literals; or a disjunction □$C$ where □ is the disjunction symbol and $C$ is a set of literals; or a negation $\neg S$ where $S$ is a sentence.

**2.1.2.4.1.** *Sets of sentences.* A set of sentences is a list of sentences enclosed in a pair of braces, exactly as in 2.1.2.2.1 above.

**2.1.3.** We will usually write an equation $=\{T_1, T_2\}$, having only two components, in the more conventional fashion as: $T_1=T_2$. Also a conjunction ■$\{S_1, \ldots, S_n\}$ will usually be written: $(S_1\wedge \ldots \wedge S_n)$ or even simply as $S_1\wedge \ldots \wedge S_n$. Likewise a disjunction will usually be written with the familiar $\vee$ interposed between its components. However the *empty* conjunction and the *empty* disjunction will always be written respectively as: ■, □, omitting the pair of braces enclosing nothing.

### 2.2. Semantics

**2.2.1.** Terms and sentences become meaningful only when an interpretation is provided for the symbols in the vocabulary over which they are written. Thereupon, as explained in detail below, each term and each sentence acquires a *denotation*, that is to say *something which it denotes*, under that interpretation. Sentences always denote one or other of the two truth values *true*, *false*. Terms always denote some specific object in the so-called *universe* of the interpretation.

**2.2.2.** *Interpretations.* Formally, an interpretation is a mapping $g$ of a vocabulary $V$ (called *the vocabulary of* the interpretation) onto a collection of entities all of which are constructed out of a certain set $D$ (called *the universe of* the interpretation). Specifically, $g$ maps each individual symbol in $V$ onto *a member of $D$*, each function symbol of degree $n$ in $V$ onto *a function from $D^n$ to $D$*, and each relation symbol of degree $n$ in $V$ onto *a function from $D^n$ to* $\{$*true*, *false*$\}$. The entity $g(E)$ onto which each symbol $E$ in $V$ is mapped by $g$ is said to be *denoted* by *$E$ under $g$* or to be *the denotation of $E$ under $g$*.

**2.2.3.** *Denotations of logical symbols, sets and sequences.* The logical symbols $\neg$, ■, □, and $=$ always denote the same entities, under every interpretation. Indeed, the negation symbol denotes that function from truth values to truth values which when applied to *true* gives *false* and conversely; the conjunction symbol and the disjunction symbol each denote a function from *sets* of truth values to truth values, with ■ denoting the function which gives *false* when applied to a set containing *false*, and giving *true* when applied to other sets; and □ denoting the function which gives *true* when applied to a set containing *true*, and giving *false* when applied to other sets. The equation symbol always denotes that function from nonempty sets to truth values which gives *true* when applied to sets containing exactly one object, and which gives *false* when applied to other nonempty sets. A set or sequence of expressions is always taken to denote the set or sequence of things which are denoted by the constituent expressions.

**2.2.4.** In general, an expression with operator $A$ and operand $B$ denotes, under an interpretation $g$ of the vocabulary in which the expression is written, the entity which the function $g(A)$ gives when it is applied to the entity $g(B)$ which is denoted by $B$ under $g$.

**2.2.5.** By virtue of the above explanations, we can regard any interpretation $g$ as automatically extended to the set of all expressions over the vocabulary of $g$. In particular each sentence over the vocabulary of $g$ denotes either *true* or *false* under $g$, and we say that the sentence is *true under* $g$, or that $g$ *satisfies* it, in the first case, and that the sentence is *false under* $g$, or that $g$ *falsifies* it, in the second case.

From our explanations above it is easy to verify that the empty conjunction is true under every interpretation and that the empty disjunction is false under every interpretation.

An interpretation can neither satisfy nor falsify a sentence which is not among the sentences over its vocabulary, for there must occur in such a sentence at least one nonlogical symbol which is without any denotation. Whenever, in the remainder of this paper, we speak of an interpretation and a sentence in the context of inquiring whether the former satisfies or falsifies the latter, we should be understood as taking it for granted that the vocabulary of the interpretation is large enough to contain each nonlogical symbol which occurs in the sentence.

**2.2.7.** It sometimes happens that an interpretation $g$ not only satisfies a sentence $S$, but that also it *would* satisfy $S$, no matter what other members of the universe of $g$ were to be denoted by the individual symbols in the vocabulary of $g$. We can express this situation more precisely with the help of the concept of *structurally equivalent* interpretations. Two interpretations are said to be *structurally equivalent* if their universes and vocabularies are the same and if each assigns the same denotation to the function symbols and relation symbols in their common vocabulary. Thus, the only way in which structurally equivalent interpretations can differ at all is in the denotations they assign to individual symbols.

Then the situation might arise that not only does $g$ satisfy $S$, but *every structural equivalent of* $g$ *satisfies* $S$.

We shall say that $g$ *strongly* satisfies $S$ if, and only if, every structural equivalent of $g$ satisfies $S$. Obviously, if $g$ strongly satisfies $S$ then $g$ satisfies $S$, because $g$ is certainly a structural equivalent of itself. But the converse is not in general true.

Intuitively, $g$ strongly satisfies $S$ only when $g$ satisfies the result of *universally quantifying* $S$ with respect to all of the individual symbols that it contains. What $S$ says about the objects denoted under $g$ by those individual symbols is true of all of the objects in the universe of $g$. In the present quantifier-free system, it is the notion of strong satisfaction which fills the gap left by doing away with the device of quantifiers.

In a similar way we say that $g$ *strongly* falsifies $S$ if, and only if, every structural equivalent of $g$ falsifies $S$. A little reflection shows that $g$ strongly falsifies $S$ only when, intuitively, $g$ satisfies the result of universally quantifying $\neg S$ with respect to its individual symbols.

In the quantifier-free predicate calculus, individual symbols are *not* variables. They always denote particular, fixed objects under an interpretation, as indeed do all nonlogical symbols.

**2.2.8.** If $X$ is a *set* of sentences and $g$ is an interpretation, we shall say that $g$ (strongly) satisfies $X$ if, and only if, $g$ (strongly) satisfies each sentence in $X$.

### 2.3. Propositions

The point of the whole enterprise of logic is to be able to formulate, investigate and settle, any *proposition* which asserts that a sentence $Y$ *follows from* a set $X$ of sentences. (To facilitate our discussion we shall say that $Y$ is the *conclusion*, and the members of $X$ the *premisses*, of the proposition $P$). Now there are two senses in which *follows from* can be taken, in our present system. The first sense, which we shall call the *ground* sense, is explained by saying that:

**2.3.1.** $Y$ *follows from* $X$ if, and only if, among the interpretations which satisfy $X$, there is none which falsifies $Y$.

The second sense, which we shall call the *general* sense, is explained by saying that:

**2.3.2.** $Y$ *follows from* $X$ if, and only if, among the interpretations which *strongly* satisfy $X$, there is none which falsifies $Y$.

In order to help keep the distinction between these two senses of *follows from* clear, we adopt the following notation for propositions: we write $X \Rightarrow Y$ to represent the proposition that $Y$ follows from $X$ in the *ground* sense, and we write $X \twoheadrightarrow Y$ to represent the proposition that $Y$ follows from $X$ in the *general* sense. We say that $X \Rightarrow Y$ is a *ground proposition* and that $X \twoheadrightarrow Y$ is a *general proposition.*

**2.3.3.** From the definitions 2.3.1 and 2.3.2 it can be readily checked that if $X \Rightarrow Y$ then $X \twoheadrightarrow Y$. This is so because the interpretations which *strongly* satisfy $X$ are all contained in the set of interpretations which merely satisfy $X$. The converse is untrue, however. For example, $\{P(x)\} \twoheadrightarrow P(y)$, but it is not the case that $\{P(x)\} \Rightarrow P(y)$.

**2.3.4.** It is not hard to show that if $X \twoheadrightarrow Y$ then any interpretation which *strongly* satisfies $X$ will also *strongly* satisfy $Y$ (although, in the definition 2.3.2, it is only formally necessary that it merely *satisfy* $Y$). For if $g$ strongly satisfies $X$, then so does $h$, where $h$ is any structural equivalent of $g$, and hence $h$ also satisfies $Y$; therefore any structural equivalent of $g$ satisfies $Y$, and therefore $g$ strongly satisfies $Y$.

**2.3.5.** A *counterexample* to a ground proposition $P$ is an interpretation which satisfies the premisses of $P$ but falsifies its conclusion. Similarly, a counterexample to a general proposition is an interpretation which *strongly* satisfies its premisses but falsifies its conclusion. Intuitively, a proposition says of itself that it has no counterexample, and it is true if this is in fact the case, and false otherwise.

A proposition which is true is also called a *theorem*.

### 2.4. Ground propositions are decidable

As we have seen, a proposition is an intuitively meaningful assertion, which says something quite concrete and specific about all of the ways in which interpretations can affect the premisses and the conclusion of the proposition. As such it is either true or false according as what it asserts to be the case *is* the case or not. Strictly speaking, a proposition is an assertion of the *semantical metalanguage* of our system, and is not to be counted among the sentences of the object language as characterized in 2.1. However, even though a proposition is always either true or false it is not always by any means obvious which. Fortunately, in the case of ground propositions, there is an *algorithmic* way of correctly deciding the truth or falsehood of them, which we now go on to explain.

**2.4.1.** *The method of denotation tables.* From the definition 2.3.1 it looks as if, in order to decide whether or not a ground proposition $P$ is true, we would have to examine all interpretations whatsoever of its premisses and conclusion. This would of course be quite out of the question, as there are at least as many interpretations of a vocabulary as there are sets to act as their universes.

However, it is not in fact necessary to do this. For the only way in which an interpretation can affect the premisses and conclusion of $P$ is by providing a truth value as denotation for each of the atoms which appear in any of them. Since there can be only finitely many (say, $n$) such atoms there can be only finitely many (at most $2^n$) different ways in which the set of them can be mapped by an interpretation onto truth values. We can list all of these ways in a *denotation table*, in just the same way as is done in constructing a *truth table* in the propositional calculus. Indeed, if we *first* construct a truth table for the set of atoms involved, and *then* remove from it any mapping which 'violates the semantics of the equation symbol' in the way explained precisely below, then we in fact get the denotation table for the set of atoms. Once we have the denotation table, we can easily check mechanically to see whether any of the mappings in it corresponds to a counterexample for $P$, and thereby settle whether $P$ is true or false. Herein consists the decision procedure for ground propositions.

**2.4.2.** *Constructing a denotation table for a set of atoms.*

**2.4.2.1.** *Conflation.* Let $T$ be a set of terms, and $K$ be a partition of $T$. Then we say that two expressions $Q$ and $R$ are *conflated by* $K$ if, for some (relation or function) symbol $S$, $Q$ is $S(A_1, \ldots, A_n)$, $R$ is $S(B_1, \ldots, B_n)$, and for each $j, j=1, \ldots, n$, $A_j$ and $B_j$ lie in the same block of $K$. Two blocks of $K$ are said to *overlap* if there are two terms, one in each of the blocks, which are conflated by $K$. By the *closure of* $K$ we mean the partition which is obtained from

$K$ by repeatedly merging (i.e., forming the union of) pairs of blocks which overlap, until there are none left which do.

2.4.2.2. *Admissible mappings.* A mapping $g$ of a set $A$ of atoms onto truth values is *admissible* if, and only if, for the partition $K$ defined below, $g$ maps atoms, which are conflated by $K$, onto the same truth value; and $g$ maps an equation $=S$ onto *true* only if $S$ is included in some block of $K$.

In the above definition, $K$ is the partition of the set $T$ of all terms which appear in any of the atoms in $A$, determined as follows: first we let $M$ be the partition of $T$ in which two terms lie in the same block if, and only if, $g$ satisfies some equation $=S$ in $A$ whose $S$ contains both of the terms: then we let $K$ be the closure of $M$.

2.4.2.3. The denotation table for a set $A$ of atoms is then the set of all mappings $g$ of $A$ onto truth values, such that $g$ is *admissible.*

### 2.5

Because of the *method of denotation tables* one can always, at least in principle, directly determine whether a ground proposition $P$ is true or not. Of course the amount of computing involved in carrying out the method on $P$ will (no matter how efficiently the work of constructing the denotation table and checking each map in it might be organized) increase as $P$ becomes 'larger'. There will indeed be a 'size' of $P$ which is as large as can feasibly be managed, using this method, for every computing agency, man or machine, with a fixed amount of computing power. Furthermore there is a point beyond which a human is not well served epistemologically by merely being *informed* that a proposition is true, or that it is false, even if his informant is entirely reliable and is somehow known to be so. One wants to be told not only *that* a proposition is true or false, but also, so to speak, *why.* It would surely be most unsatisfying intellectually to be told by an omniscient demon that, for example, Fermat's Last Theorem is indeed true, if he did not also provide us with a *proof* of it which we could understand. We go on, therefore, to discuss *inference* and *proof* in the present system and in general.

## 3. INFERRING CONSEQUENCES AND PROVING THEOREMS

### 3.1

To *prove* a proposition is to *show that it is true.* Presumably, for someone who can see directly that a proposition is true, a proof of that proposition is unnecessary. If he cannot see directly that the proposition is true, however, then he must be given a way of doing so which requires that he see directly the truth only of propositions which he *can* directly settle, without mediation. The following fundamental *consequence principle* provides the framework within which this might be done: *if Y follows from Z, and if each sentence in Z follows from X, then Y follows from X.*

It is straightforward to check that the consequence principle holds if *follows from* is construed in either the *ground* sense or the *general* sense of 2.3.

But it is important to realize that the consequence principle holds when *follows from* is simply taken in the unformalized sense of ordinary mathematical usage.

The proposition that $Y$ follows from $X$ can be seen to be true, therefore, by one who can see that $Y$ follows from $Z$ and that each member of $Z$ follows from $X$. If there are $k$ sentences in $Z$ then the original problem (to see that $Y$ follows from $X$) can be replaced by $k+1$ problems of exactly the same sort as the original one. Obviously, if the reduction is to have any point, each of the $k+1$ problems must be, in some sense, *easier to solve* than the original one. However, it is not at all necessary to treat the reduction itself as a problem, for the consequence principle requires no justification either in general or in any particular application.

These general remarks supply the motivation and background for the formal concept of *proof* which is introduced below.

**3.1.1.** A *tree of sentences* is a tree to each node of which is attached a sentence, which is said to be the sentence *at* that node. It is possible that the same sentence should be attached to more than one node in a tree of sentences. A *ground proof* is a tree of sentences such that, if $Q$ is the sentence at any interior node $N$ of the tree, and $P_1, \ldots, P_k$ are the sentences at the nodes which are immediately outward from $N$ in the tree, then the *proposition at N*, $\{P_1, \ldots, P_k\} \Rightarrow Q$, is *true*. A *general proof* is defined in exactly the same way except that $\rightarrow$ replaces $\Rightarrow$ in the definition.

Every ground proof is also a general proof, by 2.3.3, but not necessarily conversely.

If all of the sentences at the tips of the proof $P$ are in $X$, and if the sentence $Y$ is at the root of $P$, then $P$ is a proof *of* the proposition that $Y$ follows from $X$, in the ground or the general sense according as $P$ is a ground proof or a general proof.

## 3.2

In order to see, then, that a tree of sentences is in fact a proof, one must be able to see that the proposition at each of its interior nodes is in fact a *theorem*. We now describe a method which automatically produces proofs for ground theorems, in which the interior theorems are necessarily 'obvious' to the agent (man or machine) with the computing power that must have been available in order that the proof could have been produced at all. An agent having only very little computing power can produce only proofs which have interior theorems of an extremely simple kind. An agent with greater computing power can produce proofs with fewer, but 'larger' interior theorems. An agent with sufficiently great computing power will be able to prove the theorem itself in a single 'obvious' step.

## 3.3. Semantic trees

Let $K$ be the set of atoms in the premisses or the conclusion of a ground proposition $P$. Let $T$ be a tree of sentences, each one of which is a conjunction whose literals are all atoms, or complements of atoms, in $K$. Then $T$ is a *semantic tree for P* if the following four conditions are satisfied at each node $N$ of $T$, where $C$ is the conjunction of all the literals in all of the conjunctions at the nodes on the branch of $T$ down to and including $N$, and $C_1, \ldots, C_k$ are the conjunctions at the nodes of $T$ immediately outward from $N$:

**3.3.1.** there is no atom $L$ in $K$ such that both $C \Rightarrow L$ and $C \Rightarrow \neg L$;

**3.3.2.** $C \Rightarrow (C_1 \vee \ldots \vee C_k)$;

**3.3.3.** there is no literal $M$ in $C_j$ such that $C \Rightarrow M$ (for $1 \leqslant j \leqslant k$);

**3.3.4.** if $N$ is a tip of $T$, then either $C \Rightarrow L$ or $C \Rightarrow \neg L$ for each $L$ in $K$.

### 3.4. Discussion of this definition

The intuitive idea behind the definition of 3.3 is that as we move down a branch of a semantic tree for $P$ we encounter, at each node, a further quantum of information in an increasingly more complete description of an interpretation of the vocabulary in which the premisses and conclusion of $P$ are written. The conjunction $C$ of all the literals in all of the sentences on a branch of $T$ is a complete description of an interpretation in the sense that it portrays one possible way in which an interpretation $g$ can make each sentence $S$ over that vocabulary denote a truth value: if $S$ is true under $g$ then $C \Rightarrow S$, and if $S$ is false under $g$ then $C \Rightarrow \neg S$. Conditions 3.3.1 and 3.3.4 are imposed in order to ensure just this. Condition 3.3.3 is theoretically dispensable. It merely ensures that each component of each new quantum is in fact new information, not deducible from the part of the description which is already given. Condition 3.3.2 is imposed in order to guarantee that every possible interpretation *is* described by some branch of $T$. For no matter what interpretation $g$ of the premisses and conclusion of $P$ we consider, the conjunction ■ at the root of $T$ is satisfied by $g$; and in general, if $g$ satisfies $C$ then $g$ satisfies $(C \wedge C_j)$, for some $j$, by condition 3.3.2. Therefore there is some branch of $T$ which $g$ satisfies. But to say that $g$ satisfies $C$, and to say that $C$ completely describes $g$, is to say the same thing, when $N$ is a tip of $T$.

### 3.5. Failure points; counterexample trees

A *counterexample tree for P*, where $P$ is a ground proposition, is a semantic tree for $P$ in which certain nodes are classified as *failure points* as follows (retaining the notation of 3.3 and 3.4): the node $N$ in a semantic tree $T$ for $P$ is a *failure point of T* if $C \Rightarrow \neg Z$ for some premiss $Z$ of $P$ or if $C \Rightarrow Y$ where $Y$ is the conclusion of $P$.

Obviously any branch of $T$ which contains a failure point cannot describe a counterexample for $P$. Therefore, if $P$ is true, every branch of every counterexample tree for $P$ must contain a failure point.

### 3.6. Inference points of counterexample trees

A node $N$ in a counterexample tree $T$ for $P$ is called an *inference point of T* if the following two conditions are satisfied:

**3.6.1.** $N$ is not a failure point of $T$;

**3.6.2.** each of the nodes immediately outward from $N$ *is* a failure point of $T$.

### 3.7. Standard form for propositions

We obtain considerable simplification in the subsequent discussion if we are able to assume that the propositions $P$ we deal with are all in a certain standard form, namely, the form in which $P$ satisfies the two conditions:

3.7.1. the conclusion of $P$ is $\square$;

3.7.2. each premiss of $P$ is a disjunction.

There is no loss of generality involved in this assumption since every proposition is *equivalent* to a proposition in this standard form in the strict sense that every counterexample of the one is also a counterexample of the other. To see this, it is enough to note that $\{X_1, \ldots, X_n\} \Rightarrow Y$ is equivalent to $\{X_1, \ldots, X_n, \neg Y\} \Rightarrow \square$; that replacing $\neg\square\{S_1, \ldots, S_n\}$ by $\blacksquare\{\neg S_1, \ldots, \neg S_n\}$ or replacing $\neg\blacksquare\{S_1, \ldots, S_n\}$ by $\square\{\neg S_1, \ldots, \neg S_n\}$ anywhere in a proposition produces an equivalent proposition; that deletion of $\neg\neg$ anywhere in a proposition produces an equivalent proposition; and finally that replacing, on the left hand side of a proposition, the conjunction $\blacksquare\{S_1, \ldots, S_n\}$ by the $n$ disjunctions $\square S_1, \ldots, \square S_n$ produces an equivalent proposition.

### 3.8. Making inferences at inference points

Let us now examine more closely the situation at an inference point $N$ in a counterexample tree $T$ for a proposition $P$ in standard form. Let $C$, $C_1$, $\ldots$, $C_k$ be defined at $N$ as in 3.3, and let $P_1, \ldots, P_k$ be premisses of $P$ such that:

3.8.1. $\{C, C_j\} \Rightarrow \neg P_j$ for each $j$, $1 \leqslant j \leqslant k$.

Since $P_j$ is a disjunction, and since $N$ is not a failure point of $T$, we can write $P_j$ as $(A_j \vee B_j)$, where $A_j$ and $B_j$ are disjunctions, and where

3.8.2. $C \Rightarrow \neg A_j$ for each $j$, $1 \leqslant j \leqslant k$

but

3.8.3. $C \Rightarrow \neg B_j$ for no $j$, $1 \leqslant j \leqslant k$.

It is possible that $A_j$ is empty, but not that $B_j$ is. Because of 3.8.1 and 3.8.2. we have

3.8.4. $\{C, C_j\} \Rightarrow \neg B_j$ for each $j$, $1 \leqslant j \leqslant k$

and by definition of a counterexample tree we have:

3.8.5. $C \Rightarrow (C_1 \vee \ldots \vee C_k)$.

From 3.8.4 and 3.8.5 it immediately follows that:

3.8.6. $\{B_1, \ldots, B_k\} \Rightarrow \neg C$

and that therefore there is at least one choice of a disjunction $B$ (namely the disjunction of all the complements of literals of $C$) such that:

3.8.7. $\{B_1, \ldots, B_k\} \Rightarrow B$ and $B \Rightarrow \neg C$.

Now let $Q$, for any $B$ which satisfies, 3.8.7, be the disjunction:

3.8.8. $(A_1 \vee \ldots \vee A_k \vee B)$;

then it readily follows that

3.8.9. $\{P_1, \ldots, P_k\} \Rightarrow Q$

and also that

3.8.10. $C \Rightarrow \neg Q$.

For from the second part of 3.8.7 we know that $C\Rightarrow\neg B$, and this, with 3.8.2, immediately gives 3.8.10; while the first part of 3.8.7 immediately gives 3.8.9. Suppose that the proposition $P$ is: $X\Rightarrow\square$.

If we add the sentence $Q$ to the set $X$, to obtain the set $X'$, and define the tree $T'$ to be the result of classifying as extra failure points the nodes of *$T$ at which $Q$ is false*, then $T'$, as it stands, is a counterexample tree $T'$ for the proposition $X'\Rightarrow\square$. Now, since $X\subseteq X'$, every failure point of $T$ is a failure point of $T'$. However, *the node $N$ is certainly a failure point of $T'$*, because of 3.8.10, *but not of $T$*, by 3.6.1. If we define the *size* of a counterexample tree to be the number of nodes in it which are not failure points of it, then we can express the above situation by saying that *the size of $T'$ is strictly less than the size of $T$.* Now if $X\Rightarrow\square$ is true, *every* branch of $T$ contains a failure point, and therefore $T$ contains an inference point, unless the root of $T$ is itself a failure point. Hence the same will be true of $T'$. Therefore the above construction can be iterated to produce a sequence of counterexample trees $T, T', \ldots, T^{(n)}$ for a sequence $X\Rightarrow\square, X'\Rightarrow\square, \ldots, X^{(n)}\Rightarrow\square$ of theorems each of which is equivalent to $X\Rightarrow\square$ and therefore true; with the sizes of the successive trees forming a strictly decreasing sequence of numbers $\geqslant 0$. Hence, for some $n$, the size of $T^{(n)}$ will be zero. This means that the *root* of $T^{(n)}$ is a failure point for $X^{(n)}\Rightarrow\square$, which can happen only if $\square$ was inferred at some inference point in $T^{(n-1)}$, and added to $X^{(n-1)}$ to form $X^{(n)}$. If we *attach* to each of the failure points of $T$, a sentence in $X$ which is falsified at that point; and then thereafter attach, to each of the new failure points in $T'$, the sentence $Q$ which is added to $X$ to get $X'$, and so on, through the sequence to $T^{(n)}$; then $T^{(n)}$ will actually be a proof of $X\Rightarrow\square$. Each of the inferences in this proof will have been made automatically, from the materials available in the immediate neighbourhood of the corresponding inference point. Notice that a special case of a counterexample tree for any theorem $\{X_1, \ldots, X_k\}\Rightarrow\square$ is the tree having just $k+1$ nodes, namely a root $N$ and $k$ tips $N_1, \ldots, N_k$ immediately outward from $N$, with the conjunction $\blacksquare$ attached to $N$, and the conjunction $C_i$ attached to $N_i, i=1, \ldots, k$, where $C_i$ is just the conjunction of all the complements of literals in $X_i$. Then each of $N_1, \ldots, N_k$ is a failure point, and the construction of the present paragraph shows that $\square$ would be inferred directly from $\{X_1, \ldots, X_k\}$. Of course, to know that this simple tree *is* a counterexample tree for $\{X_1, \ldots, X_k\}\Rightarrow\square$ is already to know that $\{X_1, \ldots, X_k\}\Rightarrow\square$, because this is the content of 3.3.2 in this case. The upshot of this paragraph is therefore this; that from *any* counterexample tree $T$ for a theorem $X\Rightarrow\square$, *we automatically get a proof of $X\Rightarrow\square$*, in which each inference is an application of the following principle:

**3.8.11.** *from $(A_1 \vee B_1), \ldots, (A_k \vee B_k)$ one may infer the 'resolvent' $(A_1 \vee \ldots \vee A_k \vee B)$, whenever $\{B_1, \ldots, B_k\}\Rightarrow B$,* where $B$ is a *disjunction.*

It is this principle which we call the *generalized ground resolution principle.*

### 3.9. Discussion of the generalized ground resolution principle

The principle 3.8.11 specializes, in various ways, to all of the various versions of the ground resolution principle (Robinson, 1965), when $B$ is $\square$. When $B$ is not $\square$, 3.8.11 condones inferences of a rather more general character, including all those which involve the notion of equality in an essential way. It is to be noted that in applying 3.8.11 one has to *discover* a $B$ satisfying the side condition $\{B_1, \ldots, B_k\} \Rightarrow B$. In the construction of proofs described in 3.8, this discovery is done automatically, and emerges from the information available at an inference point of a counterexample tree. There is, however, a more subtle way of selecting $B$ in that construction than the one there mentioned, which we now explain.

**3.9.1.** *Selecting a B.* It is possible to make a better choice of $B$ than simply (as was indicated in 3.8) to set it equal to the disjunction of all the complements of the components of $C$. The conditions which $B$ has to satisfy, in order that the argument of 3.8 go through, are (in the notation of that argument):

**3.9.1.1.** $\{B_1, \ldots, B_k\} \Rightarrow B$ and $B \Rightarrow \neg C$.

Now it is possible to consider the set $M$ of all disjunctions whatever, in which there occur only the equality symbol, relation symbols and terms that occur in $B_1, \ldots, B_k$, and $C$. *There are only finitely many of these.* A denotation table for the set of atoms in $M$ can be constructed, and with its help, one can compute *all* of the disjunctions $B$ in $M$ satisfying 3.9.1.1, and then choose the *simplest* of these with which to construct the *resolvent* $Q$ 3.8.8. In order that this be done mechanically one must of course specify a computable measure of the simplicity of $B$, such as: the number of symbols in $B$, or the number of components in $B$.

## 4. GENERAL PROOFS

Now that we have the facility, given a ground proposition $X \Rightarrow \square$ which is true, to construct automatically a *proof* of $X \Rightarrow \square$ by using the method explained in 3.8 of converting a counterexample tree for $X \Rightarrow \square$, we go on to consider next the question of constructing, given a general proposition $X \rightarrow \square$ which is true, a *proof* of $X \rightarrow \square$.

### 4.1. Variants, instances, and substitutions

**4.1.1.** A *substitution* is an operation $\theta$ which can be performed on an expression $E$ to obtain another expression $E\theta$; the operation consists of replacing each occurrence in $E$ of each of a list $x_1, \ldots, x_n$ of distinct *individual symbols* by an occurrence of the corresponding term in a list $t_1, \ldots, t_n$ of (not necessarily distinct) *terms*. It is always assumed that $t_i$ is different from $x_i$. We write $\theta = (t_1/x_1, \ldots, t_n/x_n)$. The *empty substitution*, conventionally denoted by $\epsilon$, is the (null) operation on $E$ of replacing nothing in $E$. Thus, $E\epsilon = E$ for all $E$. The *composition* $\theta\lambda$ of two substitutions is the substitution $\mu$ such that $E\mu = (E\theta)\lambda$ for all $E$. The components of $\theta\lambda$ are easily obtained from those of $\theta$ and $\lambda$: indeed, if $\theta = (t_1/x_1, \ldots, t_n/x_n)$ and $\lambda = (u_1/y_1, \ldots, u_m/y_m)$, then

$\theta\lambda=(t_1\lambda/x_1, \ldots, t_n\lambda/x_n, u_1/y_1, \ldots, u_m/y_m)^*$ where * indicates the operation of deleting any component $t_j\lambda/x_j$ in which $t_j\lambda=x_j$, and any component $u_i/y_i$ such that $y_i$ is among $x_1, \ldots, x_n$. Composition of substitutions is associative, and $\epsilon$ is both a left and a right identity:

**4.1.1.1.** $(\theta\lambda)\mu=\theta(\lambda\mu)$ for all $\theta$, $\lambda$, $\mu$;

**4.1.1.2.** $\epsilon\theta=\theta\epsilon=\theta$ for all $\theta$.

**4.1.2.** *Instances.* An expression $Y$ is an *instance of* an expression $X$ if $Y=X\theta$ for some substitution $\theta$.

**4.1.3.** A substitution $\theta=(t_1/x_1, .., t_n/x_n)$ is *invertible*, and has the *inverse* $\theta^{-1}=(x_1/t_1, .., x_n/t_n)$, if $(x_1/t_1, .., x_n/t_n)$ is a substitution, that is, if $t_1, .., t_n$ *are distinct individual symbols.*

**4.1.4.** *Variants.* An expression $Y$ is a *variant* of an expression $X$ if $Y$ is an instance $X\theta$ of $X$ for some invertible $\theta$ such that $X=Y\theta^{-1}$.

Obviously, if $Y$ is a variant of $X$ then $X$ is a variant of $Y$; if $X$ is a variant of $Y$ and $Y$ is a variant of $Z$ then $X$ is a variant of $Z$; and $X$ is a variant of $X$.

### 4.2. Lemma

For any expressions $X_1, \ldots, X_n$ and substitutions $\theta_1, \ldots, \theta_n$, we can find variants $X_1', \ldots, X_n'$ of $X_1, \ldots, X_n$, and a substitution $\theta$, such that:

$$(X_1\theta_1, \ldots, X_n\theta_n)=(X_1'\theta, .., X_n'\theta)$$

(i.e., $X_i\theta_i=X_i'\theta$ for all $i=1, \ldots, n$).
The proof is very easy.

### 4.3

If $X$ is a *set* of expressions and $\theta$ a substitution, then by $X\theta$ we mean the *set* of all expressions $E\theta$, where $E$ is in $X$.

### 4.4

Let $X$ be a set of expressions and $\theta$ a substitution. Let $P$ be the partition of $X$ determined by the rule that $E$ and $F$ are in the same block of $P$ if and only if $E\theta=F\theta$. We say that $P$ is *induced in $X$ by $\theta$.*

### 4.5

Let $X$ be a set of expressions and $P$ a partition of $X$. We say that $P$ is a *unifiable* partition of $X$ if and only if there is some substitution $\theta$ which induces $P$ in $X$.

### 4.6

Two substitutions are said to be *equivalent over* a set $X$ of expressions if they induce the same partition in $X$.

**4.6.1.** *Comment.* There is obviously an equivalence class of substitutions over $X$ for each unifiable partition of $X$. If $X$ is finite, there are then clearly only finitely many such partitions and hence only finitely many such equivalence classes of substitutions.

### 4.7

Let $X$ be a finite set of expressions. A set $\{\theta_1, \ldots, \theta_n\}$ of substitutions is said to be a *basis* of $X$ if for each unifiable partition $P$ of $X$ there is exactly one $\theta_i$ in the set which induces $P$ in $X$.

### 4.8

*Prime bases of sets of expressions.* Let $X$ be a finite set of expressions. A basis $\{\sigma_1, \ldots, \sigma_n\}$ of $X$ is said to be a *prime* basis of $X$ if, for any basis $\{\theta_1, \ldots, \theta_n\}$ of $X$, we have

**4.8.1.** $\{\theta_1, \ldots, \theta_n\} = \{\sigma_1\lambda_1, \ldots, \sigma_n \lambda_n\}$ for some set $\{\lambda_1, \ldots, \lambda_n\}$ of substitutions.
*Comment.* Every finite set $X$ of expressions has a prime basis. Moreover, given $X$, we can *compute* a prime basis of $X$; and given a prime basis of $X$ we can *compute*, for any other basis of $X$, the substitutions $\lambda_i$ of 4.8.1. These computations are made by means of the *prime basis algorithm*, explained in the next paragraph.

### 4.9. The prime basis algorithm

Given the finite set $X$ of expressions as input, we can, for each partition $P$ of $X$, calculate the substitution $\sigma(P)$ by applying to $P$ the *unification procedure* described in 4.9.1 below.
Then let $\{\sigma_1, \ldots, \sigma_n\}$ be the set of all $\sigma(P)$ such that $\sigma(P)$ induces $P$ in $X$. This set is a prime basis for $X$.

**4.9.1.** *Unification procedure.* Given a partition $P$ of a set $X$ of expressions, we compute a substitution $\sigma(P)$ as follows:

*Step* 1. Put $\beta_0 = \epsilon$, $k=0$, and go to step 2.

*Step* 2. If $B\beta_k$ is a singleton for each block $B$ of $P$, put $\sigma(P) = \beta_k$ and *halt*. Otherwise:

*Step* 3. let $B$ be any block of $P$ such that $B\beta_k$ is not a singleton, let $E, F$ be two distinct expressions in $B\beta_k$, and let $W, Y$ be the two distinct *expressions* obtained by analyzing $E, F$ as:

$$E = AWR,\ F = AYS$$

for some (possibly empty) string $A$ of symbols and some strings (possibly empty) $R$ and $S$ of symbols.

*Step* 4. If one of $Y, W$ is an individual symbol $x$ and the other is a term $t$ in which $x$ does not occur, put $\mu_k = (t/x)$, $\beta_{k+1} = \beta_k \mu_k$, add 1 to $k$, and return to step 2. Otherwise, put $\sigma(P) = \beta_k$, and *halt*.

**4.9.2.** *Comment.* The freedom of choice (of $B, E, F$, and $x$) in steps 3 and 4 of the unification procedure will of course be removed in some fixed way in any mechanization of the procedure. For our present purposes we assume the method of choice fixed but we do not insist on any one way of fixing it. On this assumption, the sequences $\beta_0, \ldots, \beta_k$ and $\mu_0, \ldots, \mu_{k-1}$ of substitutions, generated as the procedure returns repeatedly to step 2, are fixed functions of $P$ which we call the *unification sequences for P*. It is straightforward to show that if $\theta$ is any substitution which induces $P$ in $X$, then $\sigma(P)$ induces $P$ in $X$ and that moreover $\theta=\sigma(P)\lambda$, where $\lambda=\lambda_k$, the final member of the sequence $\lambda_0, \ldots, \lambda_k$ of substitutions determined as follows: put $\lambda_0=\theta$, and then for $j\geqslant 0$, solve the equation $\mu_j\, \lambda_{j+1}=\lambda_j$ for $\lambda_{j+1}$. For then the equation $\theta=\beta_j\, \lambda_j$ is easily seen by induction to be satisfied for $j=0, \ldots, k$. For $j=0$, the equation holds because $\beta_0=\epsilon$ and $\lambda_0=\theta$. And if the equation holds for $j<k$ it must hold for $j+1$, because $\beta_{j+1}\lambda_{j+1}=(\beta_j\, \mu_j)\lambda_{j+1}=\beta_j(\mu_j\, \lambda_{j+1})=\beta_j\, \lambda_j$.

**4.10**

Now let $X_1, \ldots, X_n$ be sentences and $\theta_1, \ldots, \theta_n$ be substitutions, and consider the instances $X_1\theta_1, \ldots, X_n\theta_n$ of $X_1, \ldots, X_n$ by these substitutions. By Lemma 4.2 we can find variants $X_1', \ldots, X_n'$ of $X_1, \ldots, X_n$, and a substitution $\theta$, such that $X_1\theta_1, \ldots, X_n\theta_n=X_1'\theta, \ldots, X_n'\theta$. If $Y$ is any sentence such that $\{X_1\theta_1, \ldots, X_n\theta_n\}\Rightarrow Y$, then $\{X_1'\theta, \ldots, X_n'\theta\}\Rightarrow Y$. Let $T$ be the set of all of the terms which occur in any of the sentences $X_1', \ldots, X_n', Y$. Suppose that the sentence $Y$ has the form $Y_1t_1\theta \ldots Y_nt_n\theta Z$, where each of the terms $t_1, \ldots, t_n$ is in $T$ and none of the strings $Y_1, \ldots, Y_n, Z$ contains a term $t\theta$, with $t$ in $T$. Then put $Y'=Y_1t_1 \ldots Y_nt_nZ$, so that if we apply the substitution $\theta$ to $Y'$ we obtain the sentence $Y$ back again. Thus, $\{X_1'\theta, \ldots, X_n'\theta\}\Rightarrow Y'\theta$. However, the denotation table for $\{X_1'\theta, \ldots, X_n'\theta\}\Rightarrow Y'\theta$, and that for $\{X_1'\theta', \ldots, X_n'\theta'\}\Rightarrow Y'\theta'$, where $\theta'$ is any substitution *equivalent* to $\theta$ over $T$, are completely isomorphic; and in particular we have that $\{X_1'\sigma, \ldots, X_n'\sigma\}\Rightarrow Y'\sigma$, where $\sigma=\sigma(P)$, $P$ being the partition induced in $T$ by $\theta$. Recalling that we can find $\lambda$ such that $\theta=\sigma\lambda$, and remarking that $X_i\rightarrow X_i'\sigma$ for each $i$, we conclude that the following is the case (putting $Y'\sigma=X$):

**4.10.1.** If $\{X_1\theta_1, \ldots, X_n\theta_n\}\Rightarrow Y$ then we can find a sentence $X$ such that $\{X_1, \ldots, X_n\}\rightarrow X$ and a substitution $\lambda$ such that $Y=X\lambda$.

**4.11**

The 'lifting lemma' 4.10.1 can be used to obtain, from any ground proof of a theorem $\{X_1\theta_1, \ldots, X_n\theta_n\}\Rightarrow Y$, a general proof of a theorem $\{X_1, \ldots, X_n\}\rightarrow X$ with the property that *$Y$ is an instance of $X$*. One simply takes the given ground proof and applies 4.10.1 repeatedly, from the tips inward.

**4.12**

But a more general conclusion can be drawn from the discussion in 4.10. Let $X_1, \ldots, X_n$ be (not necessarily distinct) sentences, and let $X_1', \ldots, X_n'$ be variants of $X_1, \ldots, X_n$ no two of which have an individual symbol in common. Let $T$ be the set of all terms which occur in any of $X_1', \ldots, X_n'$, and let $S$ be a prime basis of $T$. If $\sigma$ is any substitution in $S$ and $Y'$ is any sentence, we can determine whether $\{X_1'\sigma, \ldots, X_n'\sigma\}\Rightarrow Y'\sigma$, and, if so, obtain by means of 4.10.1 a general theorem of the form $\{X_1, \ldots, X_n\}\rightarrow X$, where $X=Y'\sigma$. The general theorems which are obtainable in this way are all of the general theorems which can be obtained by applying 4.10.1 to ground theorems of the form $\{X_1\theta_1, \ldots, X_n\theta_n\}\Rightarrow Y$. We can make special use of this in order to arrive at the generalized resolution principle, in the next paragraph.

### 4.13. The generalized resolution principle

A special case of the discussion in 4.12 arises when $Y'\sigma$ comes from $X_1'\sigma, \ldots, X_n'\sigma$ by the generalized ground resolution principle 3.8.11. In this case we have a general theorem $\{X_1, \ldots, X_n\} \twoheadrightarrow X$ in which $X_i$ is $(A_i \vee B_i)$, $i=1, \ldots, n$ and $X$ is the sentence $(A_1' \vee \ldots \vee A_n' \vee B')\sigma$, where $(A_1' \vee B_1'), \ldots, (A_n' \vee B_n')$ are the variants $X_1', \ldots, X_n'$ and $B'$ is any disjunction which satisfies the condition that $\{B_1'\sigma, \ldots, B_n'\sigma\} \Rightarrow B'\sigma$.

**4.13.1.** If, therefore, we apply 4.11 to a proof of $\{X_1\theta_1, \ldots, X_n\theta_n\} \Rightarrow \square$, each inference in which is an application of the generalized ground resolution principle, we get a proof of $\{X_1, \ldots, X_n\} \twoheadrightarrow \square$, each inference in which is an application of the *generalized resolution principle* stated as follows:

**4.13.2.** *Generalized resolution principle*

> From $(A_1 \vee B_1), \ldots, (A_n \vee B_n)$ one may infer the '*resolvent*' $(A_1' \vee \ldots \vee A_n' \vee B')\sigma$, provided that $\{B_1'\sigma, \ldots, B_n'\sigma\} \Rightarrow B'\sigma$, where $(A_1' \vee B_1'), \ldots, (A_n' \vee B_n')$ are variants of the sentences $(A_1 \vee B_1), \ldots, (A_n \vee B_n)$, $\sigma$ is a member of a prime basis of the set $T$ of all terms which appear in $(A_1' \vee B_1'), \ldots, (A_n' \vee B_n')$, and $B'$, where $B'$ is a disjunction.

*Comment.* It is not necessary that $(A_1 \vee B_1), \ldots, (A_n \vee B_n)$ all be different from each other. The intention behind our formulating the principle in this way is to emphasize that the several premisses of an application of the generalized *ground* resolution principle may well include distinct instances of one and the same sentence. Such inferences, when 'lifted' to the general level, correspond to applications of 4.13.2 in which the same sentence appears more than once in the listing of the premisses. The *ground* principle 3.8.11 is simply the special case of 4.13.2 for $\sigma=\epsilon$ and $(A_i' \vee B_i')=(A_i \vee B_i)$, $i=1, \ldots, n$.

### 4.14. The completeness theorem for the generalized resolution principle

The fundamental theorem of logic, in our present notation, states that:

**4.14.1.** For any finite set $X$ of sentences: if $X \twoheadrightarrow \square$, then, for some $k \geqslant 1$ $\{X_1\theta_1, \ldots, X_k\theta_k\} \Rightarrow \square$, where $X_1, \ldots, X_k$ are in $X$ and $\theta_1, \ldots, \theta_k$ are substitutions.

From this proposition, the construction of 3.8, and 4.13.1, we obtain the *completeness theorem* for the generalized resolution principle:

**4.14.2.** For any finite set $X$ of sentences: if $X \twoheadrightarrow \square$ then there is a proof of $X \twoheadrightarrow \square$ in which each inference is an application of the generalized resolution principle.

**4.14.3.** *Comment.* Indeed, if $X \twoheadrightarrow \square$ then the *immediate* inference of $\square$, directly from $X$, is an application of the generalized resolution principle.

### 4.15

If, in the construction of 3.8, we impose further restrictions on the form of the counterexample trees which may be used, we obtain corresponding restrictions on the forms of the inferences which will be made when the counterexample trees are converted by the construction into proofs. To each such set of restrictions on the form of counterexample trees will correspond a system of logic with a single inference principle that is a correspondingly restricted form of the generalized resolution principle, and the entire argument will provide the completeness theorem for that particular system of logic.

**4.15.1.** *Pairwise resolution.* The original *resolution principle* of Robinson (1965) corresponds to the restriction that (in the notation of 3.8) we always have $k=2$ and that $C_1$, $C_2$ are always $L$, $\neg L$ for some atom $L$ in $K$. Actually this restriction gives rather more, in that the resulting system (as have all systems obtained in this way) 'has equality built in'.

**4.15.2.** *Resolution with set of support. Example* 2: If we restrict the counterexample tree for $X \Rightarrow \square$ in such a way that, for some satisfiable subset $Y$ of $X$, the negation of the conjunction $C$ at an inference node $N$ never follows from $Y$ (as may always be done) then the inferred sentence $Q$ at $N$ in the resulting proof will never follow from $Y$ alone (for $C \Rightarrow \neg Q$; and if $Y \Rightarrow Q$ we would have $Y \Rightarrow \neg C$) and will thus be a proof with *set of support* $X-Y$. In this way we obtain the systems (Wos, Carson and Robinson, 1965) in which the *set of support principle* is always observed, and we therefore have the completeness theorem for any such system.

**4.15.3.** *Clash resolution.* If all but one of $C_1, \ldots, C_k$ always contain a single literal, and the remaining one contains the complements of all of these literals, then we obtain the *clash resolution* system (Robinson, 1967), of which the system of 4.15.1 is a special case.

## REFERENCES

Robinson, J.A. (1965), A machine-oriented logic based on the resolution principle. *J. Ass. comput. Mach.*, **12**, 23–41.

Robinson, J.A. (1967), A review of automatic theorem-proving. *Annual symposia in applied mathematics XIX.* Providence, Rhode Island: American Mathematical Society.

Wos, L.T., Carson, D.F. & Robinson, G.A. (1965), Efficiency and completeness of the set of support strategy in theorem-proving. *J. Ass. comput. Mach.*, **12**, 536–41.