E-RESOLUTION: EXTENSION OF RESOLUTION TO INCLUDE THE EQUALITY RELATION

James B. Morris*
Department of Co iputer Sciences
The University of Texas at Austin

Abstract

The E-Resolution inference principle described in this paper is a single-inference logic system for the first-order predicate calculus with equality. Special axioms for equality (i.e., axioms for symmetry, reflexivity, transitivity, and substitutivity) are not required to be added to the original set of clauses. Other advantages of E-Resolution are the relatively small number of intermediate clauses which must be retained in a proof and the distinct possibility that search strategies suitable for Resolution will also be suitable for E-Resolution. Although it is not known whether or not E-Resolution is complete, this topic is currently being investigated by the author.

Key terms: automatic theorem-proving, equality substitution, E-Resolution, first-order predicate calculus with equality, theorem-proving.

I. Introduction

E-Resolution Is a technique for automatically generating proofs of theorems in the first-order predicate calculus with equality. The mechanism by which substitution of equals for equals is handled is automatically part of E-Resolution, or "built-in." The completeness of E-Resolution has not, as yet, been proven, and it is therefore not known whether or not the system is complete. However, the author is currently investigating this topic.

E-Resolution incorporates and combines some ideas similar to those used In Paramodulation, k-modulation,[10] and Sibert's thesis,[9] but is different from all three. Some of the features which enhance its usefulness are:

(a) It is not necessary to add special axioms for equality to the original set of clauses, i.e., axioms for symmetry, reflexivity, transitivity, and substitutivity.

(b) The mechanism of equality substitution is not activated until its success could help produce a resolvent. This causes the number of intermediate clauses which must be retained in the proof of a theorem to be held to a minimum.

(c) The system is a direct extension of J. A. Robinson's Resolution principle[4] and is designed so that (hopefully) search techniques developed for Resolution can be used, without extensive modification, in E-Resolution. At least this seems to be the case for set of support strategy,[11] maximal clash resolution,[5,7] and merging.[1]

This investigation was supported in part by Public Health Service Research grant GM-15769 (NIH) The system is programmed in LISP and is running on the CDC 6600 as the main portion of a theorem proofchecker for Set Theory.[2] It has been used to prove several modest theorems involving equality substitution.

The following sections serve to formally describe E-Resolution. Section II introduces definitions which will be used later. Section III develops the E-Resolution system, and Section IV illustrates E-Resolution through examples. Finally, Section V discusses search strategies.

II. Definitions

The following definitions are preparatory to the matter of defining E-Resolution. It is assumed that the reader Is familiar with the notation used in [4] and [6]. Such terms as atom, literal, clause, well-formed expression, substitution, disagreement set, Herbrand universe, etc. are defined there.

It is well known that any set of sentences in the first-order predicate calculus with equality can be converted to clausal form, i.e., an equivalent set of clauses, each of which is a disjunct of literals. In the following definitions, we borrow heavily from [8].

Equality atoms. An equality atom is an atom whose predicate symbol is the special symbol "=". Rather than writing an equality atom as "-[2]ab", we will use the more readable "a - b". Note that the predicate symbol "=" is always of degree 2. Analogous to the above, "=ab" is known as an inequality atom and will be written as "a = b". We will sometimes speak of equality literals and inequality literals. An equality literal is an equality atom or a negated inequality atom. An inequality literal is an inequality atom or a negated equality atom.

Interpretation. An interpretation I of a set S of clauses is a set of literals such that for each atomic formula F that can be formed from a predicate symbol of degree n occurring in S and n terms from the Herbrand universe Hg of S, exactly one of the literals F or F is In I. An interpretation I is said to satisfy a ground clause C if at least one literal in C is also in I, and to falsify a ground clause C if the complement of every literal in C is in I. An interpretation I is said to satisfy a non-ground clause C if it satisfies every ground instance of C over Hg, and is said to falsify a non-ground clause C if It falsifies at least one ground Instance of C over Hg. An interpretation I is said to satisfy a set S of clauses if it satisfies every clause in S, and is said to falsify a set of clauses if it falsifies at least one clause in S. Q represents the empty clause, and is not satisfied by any

interpretation.

E-interpretations. An E-interpretation of a set S of clauses is an interpretation I of S having the following properties: Let α, β, and γ be any terms in the Herbrand universe H_S of S and let L be any literal in I. Then

(1) $(\alpha = \alpha) \in I$, and

(2) If L' is the result of replacing some one occurrence of α in L by β and $(\alpha=\beta) \in I$, then $L' \in I$.

Note that if I is an E-interpretation, then we can prove the following:

(1) If $(\alpha=\beta) \in I$ then $(\beta=\alpha) \in I$, and

(2) If $(\alpha=\beta) \in I$ and $(\beta=\gamma) \in I$, then $(\alpha=\gamma) \in I$.

General E-propositions. A general E-proposition states that a set of clauses follows from some other set of clauses in the sense of the first-order predicate calculus with equality. The general E-proposition which states that the set of clauses $\{Y_1,Y_2,\ldots,Y_m\}$ follows from the set of clauses $\{X_1,X_2,\ldots,X_n\}$ is written as

$$\{X_1,X_2,\ldots,X_n\} \rightarrow_E \{Y_1,Y_2,\ldots,Y_m\}$$

and is taken to mean that if an E-interpretation I satisfies $\{X_1,X_2,\ldots,X_n\}$ then it also satisfies $\{Y_1,Y_2,\ldots,Y_m\}$. We sometimes write $S \rightarrow_E \square$ to mean that S is an E-unsatisfiable set of clauses, i.e., there is no E-interpretation which satisfies S. If there is an E-interpretation which satisfies some set of clauses S, then S is said to be E-satisfiable.

III. E-Resolution

E-Resolution is a single-inference principle from which proofs of theorems in the first-order predicate calculus, with equality, may be obtained. Given a set S of E-unsatisfiable clauses which is in clausal form, E-Resolution attempts to find a proof by the application of the following simple process:

(1) Initialize tree level bound k to zero.

(2) Is $\square$ in S? If so, a proof has been found. Otherwise, add one to k and go to step 3.

(3) Generate all E-Resolvents of S at tree level bound k, and add them to S. Go back to step 2.

There are two types of E-Resolvents. As an example of the first type, suppose that C and D are the two clauses

$$(Ps_1s_2\ldots s_n \vee A)$$

$$(\sim Pt_1t_2\ldots t_n \vee B),$$

and there exist clauses $B_1,\ldots,B_i$, which are

$$(\alpha_1 = \beta_1 \vee E_1)$$
$$\vdots$$
$$(\alpha_i = \beta_i \vee E_i),$$

in S, where $A,B,E_1,\ldots,E_i$ are disjunctions, and none of the $C,D,B_1,\ldots,B_i$ have any variables in common. If θ is a substitution such that

$$\{\alpha_1 = \beta_1,\ldots,\alpha_i = \beta_i\}\theta \rightarrow_E \{s_1 = t_1,\ldots,s_n = t_n\}\theta,$$

then

$$(A \vee B \vee E_1 \vee \ldots \vee E_i)\theta$$

is an E-Resolvent of $C,D,B_1,\ldots,B_i$, in which case, the literals

$$Ps_1s_2\ldots s_n \quad \text{and} \quad \sim Pt_1t_2\ldots t_n$$

are said to be resolved upon; in general, there are more than two literals (with the same predicate symbols) in clauses C and D which are resolved upon.

As an example of the second type of E-Resolvent, suppose that C is the clause

$$(\gamma \neq \delta \vee A)$$

and that there exist clauses $B_1,\ldots,B_i$, which are

$$(\alpha_1 = \beta_1 \vee E_1)$$
$$\vdots$$
$$(\alpha_i = \beta_i \vee E_i)$$

in S, where $A,E_1,\ldots,E_i$ are disjunctions, and none of the $C,B_1,\ldots,B_i$ have any variables in common. If θ is a substitution such that

$$\{\alpha_1 = \beta_1,\ldots,\alpha_i = \beta_i\}\theta \rightarrow_E \{\gamma = \delta\}\theta,$$

then

$$(A \vee E_1 \vee \ldots \vee E_i)\theta$$

is an E-Resolvent of $C,B_1,\ldots,B_i$, in which case the inequality literal

$$\gamma \neq \delta$$

is said to be resolved upon; in general, there are more than one inequality literals in clause C which are resolved upon.

This may be compared to Resolution[4] in which the terms in the argument lists of sets of literals $L \subseteq C$ and $M \subseteq D$ must be unified in order for the resolvent using L and M to exist. Equi-unification, however, upon finding that two terms cannot be unified ponders the question: Are the terms equal? Using a device known as the equality tree generator, all possible ways of satisfying equality of the two terms using equality literals from the current E-Resolution clause set $E^n(S)$ (n^{th} E-Resolution), is discovered. This discovery process may or may not introduce new literals as part of the E-Resolvent.

The above process is described and formalized in the definitions appearing below. Readers familiar with [4] will note a similarity between many of the definitions there and certain definitions below. This is done purposely, so that those readers will be able to understand E-Resolution more easily.

The equality tree generator (ETG). The equality tree generator is a formal device for specifying all possible ways in which a term α may be transformed into a term β by means of equality substitutions in some finite universe of equality relationships. The equality relationships are defined by equality literals in some set of clauses, C_{EQ}. Another input to ETG is an integer k, known as the tree level bound, which specifies a maximum node level bound on the equality tree. The reader is urged to refer to the example (Figure 1) as an aid to understanding the definition.

Every node in the equality tree T, generated by ETG, has a name which is an ordered triplet $\langle \gamma, C, \theta \rangle$ where γ is a term, C is a clause, and θ is a substitution. The root node of T is named $(\alpha, \square, \epsilon)$. A son node of T is formed from node $\langle \gamma, C, \theta \rangle$ as follows:

> If there is a most general unifier σ of γ and $\beta\theta$, then the only son node of $\langle \gamma, C, \theta \rangle$ is $\langle \beta\theta\sigma, C\sigma, \theta\sigma \rangle$. Otherwise, let D be a clause in C_{EQ} which has no variables in common with any other clause. Let L be an equality literal $\delta = \omega$, or $\omega = \delta$, appearing in D. Select a term ξ which occurs in γ and such that ξ and δ are most generally unifiable with most general unifier σ. Let γ' be the result of substitution of $\omega\sigma$ for a single occurrence of $\xi\sigma$ in $\gamma\sigma$. (Compare this with Paramodulation in [8].) Then the son node of $\langle \gamma, C, \theta \rangle$ associated with D,L, and ξ is
>
> $$\langle \gamma', (D-\{L\})\sigma \cup C\sigma, \theta\sigma \rangle.$$

A son node N' formed from node $N = \langle \gamma, C, \theta \rangle$ by the above-described process is said to have been generated from N. The clause D which is used to generate N' is said to be the clause associated with N'. If no clause was used from C_{EQ} in generating N', then there is no clause associated with N'.

Two nodes $\langle \gamma', C', \theta' \rangle$ and $\langle \gamma'', C'', \theta'' \rangle$ are called equivalent if γ' is identical to γ'' and C' is a variant of C''. The tree T is said to be fully-formed at node $N = \langle \gamma, C, \theta \rangle$ if no two son nodes of N are equivalent and there is no way to generate a new son node from N which is not equivalent to some existing son node of N. A node N is said to be terminal if (1) N has no son nodes and T is fully-formed at N, or (2) $N = \langle \gamma, C, \theta \rangle$ where $\langle \gamma, D, \lambda \rangle$ is an ancestor node of N, for any D and any λ. Otherwise, it is said to be non-terminal. An equality tree T is fully-formed if it is fully formed at every node in T which is not a tip node and every complete branch beginning at the root either

(a) contains k+1 nodes, where k is the tree level bound, or

(b) ends in a terminal node $\langle \gamma, C, \theta \rangle$, and there does not exist a λ such that $\beta\lambda$ is identical to γ, or

(c) ends in a node (possibly non-terminal) $\langle \beta\lambda, C, \theta \rangle$, for some λ. Nodes of this type are termed equality solution nodes. The set of clauses which are associated with all the nodes on the branch of T down to and including $\langle \beta\lambda, C, \theta \rangle$ is called the clause set associated with the equality solution node $\langle \beta\lambda, C, \theta \rangle$.

The equality tree generator produces a full-formed equality tree, given terms α and β, tree level bound k and set of clauses C_{EQ}.

As an example of the operation of ETG, suppose $\alpha = f(a)$, $\beta = g(h(c))$, and C_{EQ} is the set of clauses $\{\{f(x) = g(x), Qx\}, \{a = c, Pz\}, \{w = h(w)\}\}$. Then a portion of the equality tree resulting from the α, β and C_{EQ} given appears as Figure 1. The nodes marked T are terminal; those marked F are not yet fully-formed; those marked E are equality solution nodes, and those unmarked are fully-formed nodes.

Equi-unification algorithm. The equi-unification algorithm is the key to E-Resolution. It is applicable to any finite, nonempty set A of well-formed expressions, set of clauses C_{EQ} containing equality literals, and tree level bound k.

Briefly, an explanation of the operation of the algorithm is as follows: (For this explanation, we shall assume that the set A consists of two literals, $Ps_1s_2 \ldots s_n$ and $Pt_1t_2 \ldots t_n$.) Start with $E = \emptyset$. For each (s_i, t_i) pair, the algorithm attempts to find a most general unifier λ which will make $s_i\sigma\lambda = t_i\sigma\lambda$, where σ is a substitution resulting from the unifiable (s_j, t_j) pairs, for all $j < i$. If this is not possible, however, it adds the inequality $s_i'\sigma \neq t_i'\sigma$ (where s_i' and t_i' are distinct, corresponding terms occurring in s_i and t_i) to the set E and makes a substitution of $t_i\sigma$ for $s_i\sigma$ in A, only for that occurrence of $s_i\sigma$ in $Ps_1s_2 \ldots s_n\sigma$. When the above process is completed for all pairs (s_i, t_i), the set E is examined. If $E = \emptyset$, then A is most generally unifiable in the sense of [4]. However, if $E \neq \emptyset$, then the set $E\sigma_A$ is processed as described below. Note the use of $E\sigma_A$ rather than E, where σ_A is the substitution obtained from the pairs (s_i, t_i) which were unifiable.

Now, for each $\alpha \neq \beta$ in $E\sigma_A$, ETG is activated with $\alpha, \beta, E^{k-1}(S)$ and k as inputs. If for some $\alpha \neq \beta$ in $E\sigma_A$, ETG cannot find a way to show that $\alpha = \beta$, at tree level bound k, then the algorithm fails to find an equi-unifier at tree level bound k and terminates; otherwise, A is equi-unifiable.

The equi-unification algorithm is defined to be

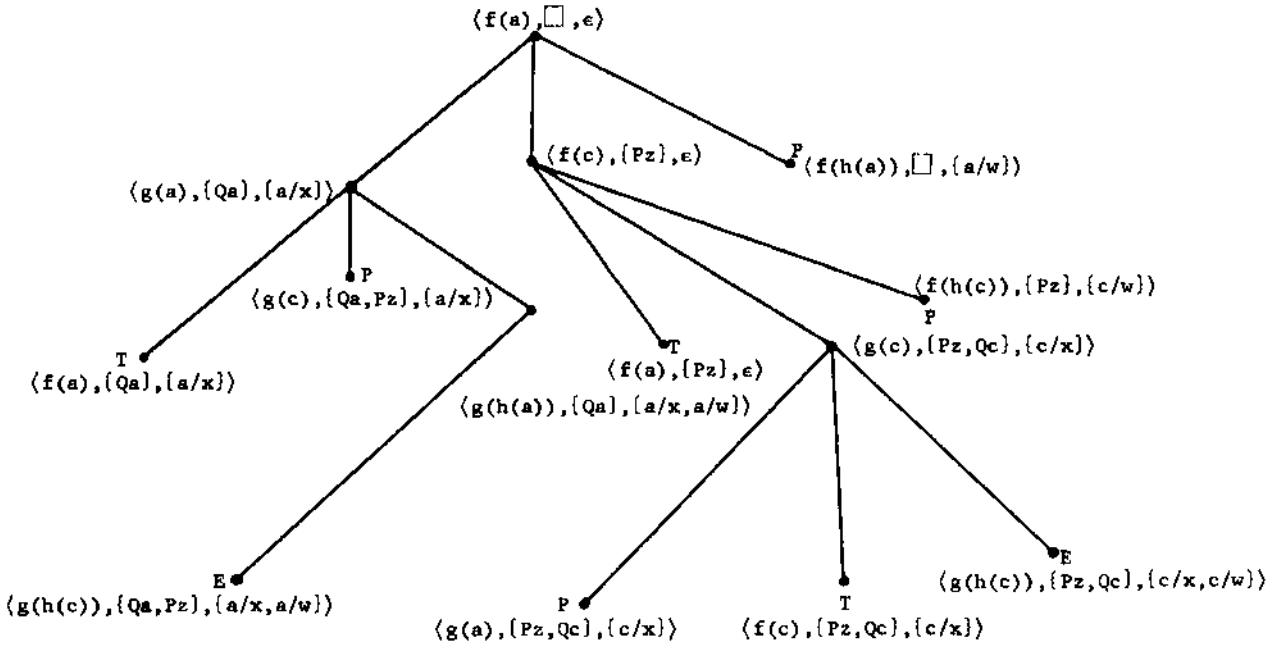


Figure 1

Step 1. Set $\sigma_0 = \epsilon$, $i = 0$, $E = \emptyset$, $P = \emptyset$, and go to Step 2.

Step 2. If $A\sigma_i$ is not a singleton, go to Step 3. Otherwise, set $E = E\sigma_i$ and go to Step 4.

Step 3. Let V_i be the earliest, and U_i the next earliest, in the lexical ordering of the disagreement set B_i of $A\sigma_i$. If V_i is a variable, and does not occur in U_i, set $\sigma_{i+1} = \sigma_i\{U_i/V_i\}$, add 1 to i and return to Step 2. Otherwise, set $E = E \cup \{V_i \neq U_i\}$, substitute U_i in $A\sigma_i$ for just that single occurrence of V_i in the disagreement set B_i and return to Step 2.

Step 4. If $E = \emptyset$, then set $\sigma_A = \sigma_i$ and terminate (equi-unifiable). Otherwise, set L equal to one of the inequalities, $V_j \neq U_j$, in E, set $E = E - \{L\}$, and activate the equality tree generator with V_j and U_j as the α and β arguments, k as tree level bound, and the set $E^{k-1}(S)$ (see below) as the argument C_{EQ}, where $E^{k-1}(S)$ is the current E-Resolution set. Go to Step 5.

Step 5. If the fully-formed equality tree T produced by ETG has no equality solution nodes, then terminate (failure to equi-unify). Otherwise, select a node $\langle\gamma,\beta,\theta\rangle$ from the finite set of all equality solution nodes in T, set $P = P \cup \{\langle\gamma,\beta,\theta\rangle\}$, set $E = E\theta$ and return to Step 4.

Partial equi-unifiers. If A is a finite, non-empty set of well-formed expressions for which the equi-unification algorithm terminates in Step 4, the substitution σ_A available as output is called the partial equi-unifier of A, and A is said to be equi-unifiable. The set P available as output from the equi-unification algorithm is termed the equality solution node set. The union of all the clause sets which are associated with each of the nodes in the equality solution node set is called the equality solution clause set associated with P.

Key quintuples. The ordered quintuple $\langle L,M,N,B,E\rangle$ is said to be a key quintuple of the ordered pair of clauses $\langle C,D\rangle$ if the following conditions are satisfied:

(1) L and M are non-empty sets of literals (which do not involve either the "=" or "≠ predicate symbols), and $L \subseteq C$, $M \subseteq D$ where C and D have no variables in common with any other clause.

(2) N is the set of atomic formulas which are members, or complements of members, of the set $L \cup M$.

(3) N is equi-unifiable, with partial equi-unifier σ_N, $P = \{\langle\gamma_1,B_1,\theta_1\rangle, \langle\gamma_2,B_2,\theta_2\rangle, \ldots, \langle\gamma_n,B_n,\theta_n\rangle\}$ is an equality solution node set available as output from the equi-unification algorithm, $B = B_1 \cup B_2 \cup \ldots \cup B_n$, and

$E = \{E_1, E_2, \ldots, E_m\}$ is the equality solution clause set associated with P.

(4) If θ is the substitution $\theta_1\theta_2\ldots\theta_n$, where the θ_i are those mentioned in (3) above, then either $N\sigma_N\theta$ is a singleton, or corresponding distinct terms occurring in the argument lists of all the literals in $N\sigma_N\theta$ are term components of the node names which are at the root node and equality solution node $\langle \gamma_i, B_i, \theta_i \rangle$ of some branch of some equality tree. Also, every member of $L\sigma_N\theta$ has the same logical sign, every member of $M\sigma_N\theta$ has the same logical sign and no two members of $L\sigma_N\theta$ and $M\sigma_N\theta$, respectively, have the same logical sign. The substitution $\sigma_E = \sigma_N\theta$ is termed the equi-unifier of the set N.

Key quadruples. The ordered quadruple $\langle M,N,B,E \rangle$ is said to be a key quadruple of the clause C if the following conditions are satisfied:

(1) M is a set consisting entirely of inequality literals occurring in C, where C has no variables in common with any other clause.

(2) If M is $\{\alpha_1 \neq \beta_1, \ldots, \alpha_i \neq \beta_i\}$, then N is the set $\{\neq(\alpha_1,\ldots,\alpha_i), \neq(\beta_1,\ldots,\beta_i)\}$.

(3) N is equi-unifiable, with partial equi-unifier σ_N, $P = \{\langle \gamma_1, B_1, \theta_1 \rangle, \langle \gamma_2, B_2, \theta_2 \rangle, \ldots, \langle \gamma_n, B_n, \theta_n \rangle\}$ is an equality solution node set available as output from the equi-unification algorithm, $B = B_1 \cup B_2 \cup \ldots \cup B_n$, and $E = \{E_1, E_2, \ldots E_m\}$ is the equality solution clause set associated with P.

(4) If θ is the substitution $\theta_1\theta_2\ldots\theta_n$, where the θ_i are those mentioned in (3) above, then either $N\sigma_N\theta$ is a singleton, or distinct corresponding terms occurring in the argument lists of the literals in $N\sigma_N\theta$ are term components of the node names which are at the root node and equality solution node $\langle \gamma_i, B_i, \theta_i \rangle$ of some branch of some equality tree. The substitution $\sigma_E = \sigma_N\theta$ is termed the equi-unifier of the set N.

E-resolvents. An E-resolvent of the clauses $C, D, E_1, E_2, \ldots, E_m$ is any clause of the form

$$(C-L)\sigma_E \cup (D-M)\sigma_E \cup B\sigma_E$$

where $\langle L,M,N,B,E \rangle$ is a key quintuple of $\langle C,D \rangle$, or an E-resolvent of the clauses $C, E_1, E_2, \ldots, E_m$ is any clause of the form

$$(C-M)\sigma_E \cup B\sigma_E$$

where $\langle M,N,B,E \rangle$ is a key quadruple of C.

E-resolvent at tree level bound k. An E-resolvent at tree level bound k is any E-resolvent produced by restricting the equi-unification algorithm to tree level bound k.

E_k-resolvent sets. The E_k-resolvent set of clauses $C, D, E_1, E_2, \ldots, E_m$ (or $C, E_1, E_2, \ldots, E_m$) is just the set of all possible, distinct E-resolvent at tree level bound k of $C, D, E_1, E_2, \ldots, E_m$ (or $C, E_1, E_2, \ldots, E_m$). Note that for any sets of literals $L \subseteq C$ and $M \subseteq D$, it is possible to generate more than one E-resolvent at tree level bound k, since we may call the equi-unification algorithm more than one time for each set of literals L and M, each time selecting different equality solution nodes, as specified in Step 5 of the algorithm, to obtain different equality solution node sets. However, there are only finitely many ways to select the nodes and eventually a point will be reached where, for a given L and M, we cannot generate any new, unique E-resolvents at tree level bound k. In a practical system, it is fairly simple to organize the program in such a way that the equi-unification algorithm is actually called only once for a given L and M, while still obtaining all of the possible E-resolvents at tree level bound k.

E_k-resolution. If S is any set of clauses, then the E_k-resolution of S denoted by $E_k(S)$, is the union of S with the union of all E_k-resolvent sets of members of S.

N^{th} E-resolution. The n^{th} E-resolution of S, where S is any set of clauses, is denoted by $E^n(S)$, $n \geq 0$, and is defined to be S, if $n = 0$, and $E_n(E^{n-1}(S))$ otherwise.

IV. Examples

The first two examples are trivial examples which illustrate interesting points about the system, and the third example is a theorem from Set Theory.[3]

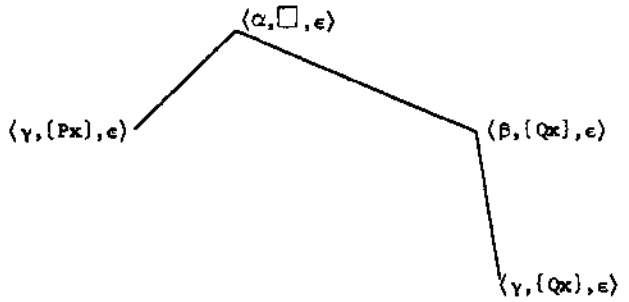
Example 1. Given the clauses

(1) $\{\alpha = \beta, Qx\}$
(2) $\{\beta = \gamma\}$
(3) $\{\alpha = \gamma, Px\}$
(4) $\{R\alpha\}$
(5) $\{\sim R\gamma\}$
(6) $\{\sim Qx\}$

From (4), (5), (1), and (2) and the key quintuple

$(\{R\alpha\}, \{\sim R\gamma\}, \{R\alpha, R\gamma\}, \{\langle \gamma, \{Qx\}, \epsilon \rangle\}, \{\{\alpha=\beta, Qx\}, \{\beta=\gamma\}\})$

where the equality-tree generated is:



we obtain

(7) {Qx}.

Note that it is also possible to conclude (by choosing a different key quintuple)

(7') {Px}.

From (6) and (7) it is obvious that we conclude

(8) □.

The above example illustrates what happens if there is more than one way of proving two terms equal. Note that if E-resolution had been designed so that it assumed only one way of proving α and β in the above set equal, the resolvent selected might possibly by (7'), in which case □ cannot be concluded. Unless the path α = β = γ is recognized, it is impossible to deduce □.

Example 2. (This example appears in a footnote in [8]. Suppose we are given the clauses

(1) {a = b, Qc}

(2) {g(a) ≠ g(b), Qc}

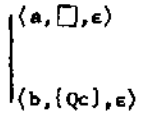
(3) {c = d, ~Qc}

(4) {g(c) ≠ g(d), ~Qc}.

This set of clauses is E-unsatisfiable and illustrates the necessity of those E-resolvents which are designed to remove inequality literals from clauses. Note that if we apply only those E-resolvents which are associated with key quintuples, we cannot deduce □ from this set of clauses. We must also utilize those E-resolvents which are associated with key quadruples. From (1) and (2) with key quadruple

⟨{g(a)≠g(b)}, {≠(g(a)),≠(g(b))}, {⟨b,{Qc},ε⟩}, {{a=b,Qc}}⟩

and the equality tree



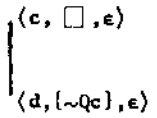
we obtain

(5) {Qc}.

Similarly, from (3) and (4) with key quadruple

⟨{g(c)≠g(d)}, {≠(g(c)),≠(g(d))}, {⟨d,{~Qc},ε⟩}, {{c=d,~Qc}}⟩

and equality-tree



we obtain

(6) {~Qc}.

From (5) and (6) we obtain

(7) □.

Example 3. The following example is Theorem 2.40.3 in [3]. The theorem is stated as:

((a ∈ b ∩ sng a ∈ U) = (a ∈ b)).

Previously proved theorems in [3] tell us

(a) Theorem 2.24.6 (x ∩ U = x)

(b) Theorem 2.24.8 (0 ∩ x = 0)

(c) Theorem 2.24.50 (x ∈ y → (x ∈ y) = U)

(d) Theorem 2.24.51 (~(x ∈ y) → (x ∈ y) = 0)

(e) Theorem 2.12 (x ∈ y → x ∈ U)

(f) Theorem 2.34.5 (x ∈ U → sng x ∈ U)

where "sng" is the set operator which converts its one argument into a singleton set, e.g., sng a = {a}.

The clauses numbered 1-6 below come from Theorems (a)-(f) above. Clause 7 comes from the negation of the theorem, and clauses 8-12 denote the clauses obtained in the proof of the theorem. For convenience, we denote "x ∩ y" by f(x,y), "x ∈ y" by the predicate Q(x,y) or the term g(x,y), "sng x" by h(x) and write the equality predicates in the more familiar manner. Also, U is the constant denoting the universe, and 0 is the constant denoting the empty set.

(1) {f(x,U) = x}

(2) {f(0,x) = 0}

(3) {Q(x,y), g(x,y) = 0}

(4) {~Q(x,y), g(x,y) = U}

(5) {~Q(x,y), Q(x,U)}

(6) {~Q(x,U), Q(h(x),U)}

(7) {f(g(a,b), g(h(a),U)) ≠ g(a,b)}

(8) {g(x,y) = 0, Q(x,U)} from (3) and (5)

(9) {g(x,y) = 0, Q(h(x),U)} from (6) and (8)

(10) {g(x,y) = 0, g(h(x),U) = U} from (4) and (9)

(11) {g(a,y) ≠ 0} from (7), (1), and (10)

(12) □ from (7), (2), and (11).

VI. Search Strategies

As stated earlier, it appears that set of support strategy, maximal clash resolution, and merging, as well as other search strategies suitable for Resolution may all be used without extensive modification in obtaining E-Resolution proofs. For example, maximal clash resolution is used in the theorem proof-checker discussed in [2].

The reasoning behind the statements in the above paragraph is simple. Note that E-Resolution proceeds exactly as does Resolution until (in Resolution) two terms cannot be unified. At this point, E-Resolution makes a divergence. It checks to see if indeed these two terms which could not be unified, can be equi-unified. If this is the case, then a (possibly empty) set of literals, which are the by-product of equi-unification, will be unioned with the result of resolving (in the straight Resolution sense) the two clauses currently under investigation. Regardless of whether the two terms are equi-unifiable or not, E-Resolution begins again at exactly the same point where the divergence was initiated and proceeds from this point exactly as would Resolution, dependent upon whether the terms were unifiable or not. The only real difference is the set of by-product literals, which accumulate as a result of equi-unification, in E-Resolution.

Although no proof is given of the assertion, it seems true that for many of the same reasons that the above-mentioned strategies do not destroy completeness in Resolution, they do not destroy completeness in E-Resolution.

Summary

E-Resolution may be compared to other methods for handling equality substitutions in Resolution, namely, Demodulation, 10 Paramodulation, and the system described by Sibert.9

The Demodulation system is the forerunner of the more general, more powerful Paramodulation system. In the generating of equality trees by ETG, a portion of the technique used is very similar to Paramodulation and Demodulation and in fact was inspired by the basic techniques of these two systems. However, in E-Resolution the only equality substitutions which result in a new clause are those which will produce an E-resolvent, although many intermediate substitutions may be made in order to find the E-resolvent. In Paramodulation, by contrast, equality substitutions may result in new clauses which will not produce resolvents. This is the concept in E-Resolution which results in generation of only a minimum number of new clauses.

The Sibert system is more similar to E-Resolution than are Paramodulation and Demodulation. However, it is felt by the author that E-Resolution is a more efficient system and, aa such, in its present form is more practical for use on a machine. However, many of the concepts introduced by Sibert are basic to the problem of Resolution with equality and were used as a foundation for E-Resolution.

A disadvantage of E-Resolution and an area in which further research would no doubt increase greatly the efficiency of a system utilizing it, is the time which must be spent by ETG fruitlessly constructing an equality tree when, in fact, the *a* and B terms given it are not equal. In general, this condition will actually occur more times than will the condition where they are equal. However, the following process will rectify this problem: the tree level bound k may be set to some low level initially. A bound may also be set on the number of E-resolvent sets which may be generated, i.e., a maximum n such that $E^{n+1}(S)$ is not generated. Now, in generating $E^n(S)$, one saves all equality trees associated with non-equi-unifiable literals of level k. If Q is not in $E^n(S)$, then one increases k to k' and restarts the E-Resolution process, each time extending and saving the equality trees associated with non-equi-unifiable literals at level k'. If [J is not in $E^n(S)$, raise the maximum E-resolution level n to n' and generate E^n (S). If Q] is not In E^n (S), then k' is increased to k", etc.

Acknowledgements. The author would like to express appreciation to Dr. W. W. Bledsoe for much guidance, encouragement, and consultation given to him throughout the preparation of the E-Resolution system and the material presented here. The author would also like to acknowledge the valuable consultation given to him during his research efforts by Messrs. Robert Anderson and Forest Baskett.

References

1. Andrews, Peter B., Resolution with Merging, J. ACM 15, 3 (July, 1968), 367-381.
2. Bledsoe, W. W., Theorem Proof Checking in Set Theory, in preparation.
3. Morse, A. P., A Theory of Sets, Academic Press, New York, 1965.
4. Robinson, J. A., A Machine-oriented Logic Based on the Resolution Principle, J. ACM 12, (Jan., 1965), 23-41.
5. Robinson, J. A., Automatic Deduction with Hyper-Resolution, Int. J. Comp. Math. 1, 1965, 227-234.
6. Robinson, J. A., The Generalized Resolution Principle, In Machine Intelligence III, D. Michie (ed.), Edinburgh University Press, 1968, 113-127.
7. Robinson, J. A. A Review of Automatic Theorem Proving, Proc. Symp. in Applied Mathematics, Amer. Math. Soc., Providence, R. I., 1967.

8. Robinson, G., and L. Wos, Paramodulation and Theorem-proving in First-order Theories with Equality. To appear in Machine Intelligence IV, D. Michie (ed.)

9. Sibert, E. E., A Machine-oriented Logic Incorporating the Equality Relation, Doctoral Thesis, Rice University, Houston, 1967.

10. Wos, L., G. Robinson, D. Carson, and L. Shalla, The Concept of Demodulation in Theorem-proving, J. ACM 14 (Oct., 1967), 698-709.

11. Woe, L., G. Robinson, and D. Carson. Efficiency and Completeness of the Set of Support Strategy in Theorem-proving, J. ACM 12 (1965), 536-541.