AND-OR GRAPHS APPLIED TO RUE RESOLUTION

Vincent J. Digricoli
Dept. of Computer Science
Fordham University
Bronx, New York 104-58

James J, Lu, V. S. Subrahmanian
Dept. of Computer Science
Syracuse University-
Syracuse, New York 13210

ABSTRACT

In equality-based binary resolution, the viability test is used as a decision mechanism to select disagreement sets and also to define the RUE unifier. In this paper we solve the problem of non-termination of the viability test by applying the theory of And-Or graphs.

RUE refutations are succinct and typically less than half as long as corresponding refutations with the equality axioms. Furthermore, RUE resolution is complete without using paramodulation or the equality axioms. Experiments with a theorem prover based on this method have produced superior results which are described in [Digricoli and Harrison,1986].

1. INTRODUCTION

In resolution by unification and equality (RUE resolution), we augment the standard theory of binary resolution due to [Robinson,I965] by the notion of disagreement sets, to incorporate the axioms of equality into the definition of resolution and eliminate the use of these axioms in refutations.

In RUE resolution we resolve to inequalities when the mgu does not exist; for example, the E-unsatisfiable set:

S: 1. $\overline{P}(f(a),g(b))$
2. $\overline{P}(f(c),g(d))$
3. $a = b$
4. $b = c$
5. $g(c) = g(d)$

is refuted by:

$P(f(a),g(b))$

$\vdash\!\!-\!\!-\!\!-$ $\overline{P}(f(c),g(d))$

$a \neq c \;\vee\; g(b) \neq g(d)$

$\vdash\!\!-\!\!-\!\!-$ $g(c) = g(d)$

$a \neq c \;\vee\; b \neq c$

$\vdash\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-$ $a = b$

$b \neq c$ (merging)

$\vdash\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-$ $b = c$

□

Acknowledgement: Subrahmanian supported by USAF Grant F30602-85-C-0008.

2. RUE Resolution

The basic definitions underlying RUE resolution are the following:

A Disagreement Set of a Pair of Terms:

"If s,t are non-identical terms, the set of one element, a pair, [s:t], is the origin disagreement set. If s,t have the form: $f(a_1,\ldots,a_k), f(b_1\ldots,b_k)$, then the set of pairs of corresponding arguments that are not identical is the topmost disagreement set. Furthermore, if D is a disagreement set, then (D - [e]) U D is also a disagreement set, where e is an element of D and D is a disagreement set of e."

For example:

{f(a,g(b)):f(c,g(d))} origin dis. set
{a:c, g(b):g(d)} topmost
{a:c, b:d} bottommost

A Disagreement Set of Complementary Literals:

"A disagreement set of complementary literals, $P(s_1,\ldots,s_n), P(t_1,\ldots,t_n)$, is defined as the union:

$$D = \bigcup_{i=1}^{n} D_i$$

where D_i is a disagreement set of the corresponding arguments, s_i, t_i .

The topmost disagreement set of $P(s_1,\ldots,s_n)$, $\bar{P}(t_1,\ldots,t_n)$ is the set of pairs of corresponding arguments that are not identical. Applying the substitution axiom for predicates, we can state that:

$$P(s_1,\ldots,s_n) \wedge \bar{P}(t_1,\ldots,t_n) \rightarrow D$$

where D now represents a disjunction of inequalities specified by any disagreement set of $\bar{P}$, not P. In resolution by unification and equality we can resolve P and $\bar{P}$ immediately to D. For example, we may resolve:

$$P(f(a),g(b)) \quad \bar{P}(f(c),g(d)) \vdash D$$

in four distinct ways dependent on our choice of disagreement set; namely to any one of the following resolvents:

$$\begin{array}{lcl} f(a)\neq f(c) & v & g(b)\neq g(d) \\ a\neq c & v & g(b)\neq g(d) \\ f(a)\neq f(c) & v & b\neq d \\ a\neq c & v & b\neq d \end{array}$$

It is in fact, the second resolvent which is required to refute S in the previous example. Hence, RUE resolution requires a decision mechanism to select the appropriate disagreement in resolving complementary literals. Furthermore, we show in [Digricoli and Harrison,1986] that the substitution to be applied before resolving in a refutation may be neither the mgu (when it exists) nor the mgpu, the most general partial unifier, a relaxed form of the mgu which permits us to pass over irreducible disagreements. Instead we define the RUE unifier as the appropriate substitution.

In the theory of RUE resolution in strong form, both the decision mechanism to select disagreement sets and the definition of the RUE unifier depend on a viability test defined as follows:

> "A Disagreement Set D is Viable in respect to a set of clauses S only if:
>
> for each $s_i \neq t_i$ corresponding to D, we have that for s_i there is a term $\underline{a}$ which is the argument of a positive equality literal in S such that s_i either unifies with $\underline{a}$ or matches $\underline{a}$ on leading function symbol. In the latter case, s_i:a must have a viable disagreement set below the origin disagreement. Similarly for t_i and some other term $\underline{b}$ which is the argument of a positive equality literal in S."

For example, in respect to:

S: 1. $P(f(a),g(b))$
2. $\bar{P}(f(c),g(d))$
3. $a = b$
4. $b = c$
5. $g(c) = g(d)$

the complementary literals, P and $\bar{P}$, have only one viable resolvent:

$$a\neq c \quad v \quad g(b)\neq g(d) .$$

$a\neq c$ is viable since both a and c unify with arguments of equality literals in S. $g(b)\neq g(d)$ is viable since g(b) matches on function symbol with g(c) in clause 5 (with b:c being viable). Furthermore, g(d) unifies with g(d) in the same clause. Hence, the entire resolvent is viable and leads to the refutation already stated.

Notice, however, that: $a\neq c \quad v \quad b\neq d$ is not viable as a resolvent of $P,\bar{P}$, since the inequality $b\neq d$ is not viable since d does not unify with the argument of an equality literal in S. Furthermore, $f(a)\neq f(c) \quad v \quad g(b)\neq g(d)$ is not viable since $f(a)\neq f(c)$ is not viable.

By viable we mean "can possibly participate in a refutation". In [Digricoli and Harrison,1986] we prove that if an inequality is not viable, it either cannot participate in a refutation of S or must be reduced to a lower disagreement. We also state that the definition of viability is recursive and the recursive iteration is finite, since each of the terms s_i and t_i has a finite nesting of function symbols. The recursion peels these off and thus terminates.

It is precisely here that the definition of viability is flawed since the last statement is not true. Lu and Subrahmanian [1988] point out counter examples which show that a non-terminating iteration can occur in applying the viability test. The following clause set, for example, leads to non-terminating iteration:

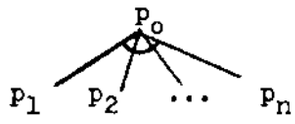
S: 1. $P(f(a,b))$
2. $\bar{P}(d)$
3. $a = b$
4. $b = c$
5. $f(f(a,b),c) = d$

P and $\bar{P}$ can only resolve to $f(a,b)\neq d$, but this inequality when tested for viability leads to a non-terminating iteration: f(a,b) matches on function symbol with f(f(a,b),c) and we must test $a\neq f(a,b)$ and $b\neq c$ for viability. But the first inequality reintroduces f(a,b) which again must be matched with f(f(a,b),c), ad infinitum.

Since in RUE theory, the concept of viability plays a central role in defining the RUE unifier and the topmost viable disagreement set, the possibility of non-termination undermines the theory of RUE resolution in strong form unless we can show how to avoid non-termination. The primary subject of this paper is to show how the theory of And-Or graphs due to [Chang and Slagle,197l] can be used to solve this problem.

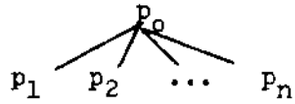
3. The Theory of And-Or Graphs

A node in an-And-Qr graph represents a problem to be solved. We define an And-expansion:



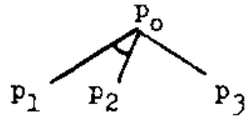
as denoting that problem p_o is solved if the logical And of subproblems $p_1,p_2,...,p_n$ is solved, i.e., we decompose p_o in to subproblems $p_1,p_2,...,p_n$, all of which must be solved in order to solve p_o.

Similarly we define an Or expansion:



to denote that p_o is solved if any one of the subproblems $p_1,p_2,...,p_n$ is soved.

Mixed expansions are permitted so that:

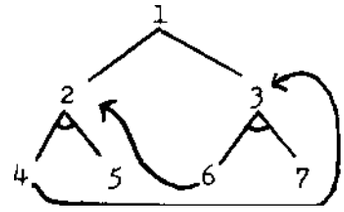


denotes that p_o becomes $(p_1 \wedge p_2) \vee p_3$.

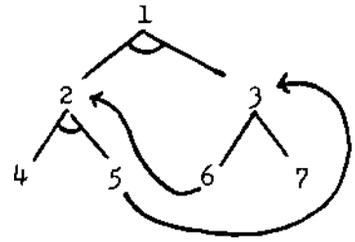
In an And-Or graph we repeatedly perform decomposition until we reach primitive subproblems which are known to be solvable or unsolvable. These are the leaf nodes .of the graph. The entire graph represents a system of boolean substitutions for the root node.

A set of leaf nodes, !the solution of all of which implies the solution of the root node, is called an implicant of the root. One implicant is said to subsume another implicant if it is a subset of that implicant. For example, if we denote nodes by numeric indices, then {7,8} subsumes {2,7,8,9} and, furthermore, since {7,8) is a simpler solution, we should discard {2,7,8,9}. Subsumption will play an important role in our work.

The situation becomes interesting because of circularity. The And-Or graph:



has no solution due to circularity. But the cyclic And-Or graph:



has one solution: $p_4 \wedge p_7$.

An And-Or graph with many nodes and cycles may represent a very complex problem decomposition. The following algorithm stated by Chang and Slagle will compute the implicants of an arbitrary And-Or graph:

Implicant Algorithm;

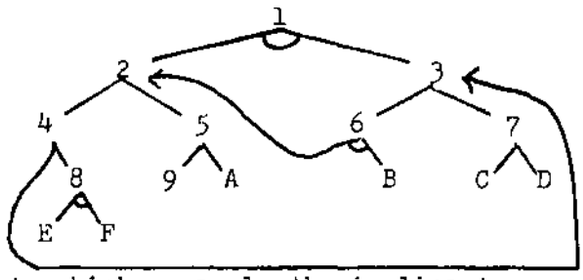
1. Initialize the current formula to the root node.

2. Select any non-leaf node in the current formula (for example, the lowest numbered node) and substitute its boolean expansion as stated in the And-Or graph, for each occurrence of this node in the current formula.

3. Remove from the current formula any implicant which is subsumed by another implicant in the current formula or by an implicant which previously appeared in the current formula.

4. Repeat steps (2) and (3) until the current formula either vanishes (there is no solution for the root due to cycles in the graph) or only leaf nodes appear in the current formula. In the latter case, the final state of the current formula, when no further substitutions are possible, represents a disjunction of the alternative implicants of the And-Or graph.

For example, consider the And-Or graph taken from [Nilsson,1971f p.126] :



to which we apply the implicant algorithm:

```
                    1
                   23
                43 v 53
             46 v 47 v 56 v 57
       36 v 86 v 37 v 87 v 56 v 57
6 v 76 v 86 v 67 v 7  v 87 v 56 v 57
                 6 v 7    (subsumption)
                2B v 7
             4B v 5B v 7
       3B v 8B v 5B v 7
6B v 7B v 8B v 5B v 7
             8B v 5B v 7    (subsumption)
       8B v 9B v AB v 7
       8B v 9B v AB v C v D
     EFB v 9B v AB v C v D all leaf nodes
```

The algorithm converges to five distinct implicants and subsumption was used to filter out the effect of the cycle in the graph.

Chang and Slagle prove that their algorithm always converges to solution implicants or to the empty formula, irrespective of cycles in the graph. Nilsson who gives a good treatment of And-Or graphs in [1971] fails to mention the implicant algorithm which seems to be absent from most texts in artificial intelligence.

4. And-Or Graphs Applied to Viability

Consider our previous clause set:

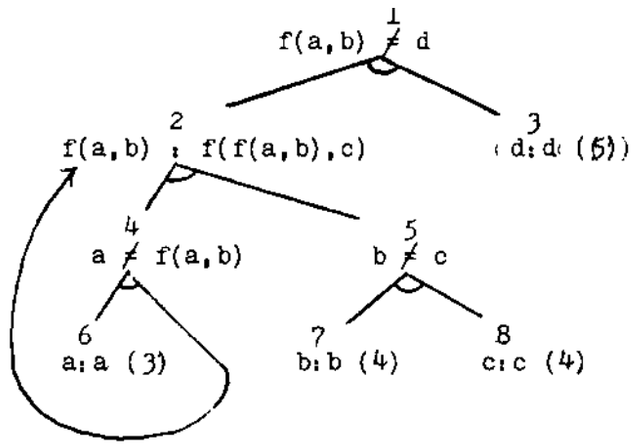
S: 1. $P(f(a,b))$
2. $\overline{P}(d)$
3. $a = b$
4. $b = c$
5. $f(f(a,b),c) = d$

Can S be refuted? The only possible resolution is:

$$P(f(a,b)) \quad \overline{P}(d)$$
$$\vdash$$
$$f(a,b) \neq d$$

so the question arises: is this inequality viable? We have previously shown that the viability test does not terminate when applied to this inequality. In point of fact, this inequality is not viable since we cannot deduce f(a,b)=d from S.

If we state the recursive application of the viability test as an And-Or graph, we can show that f(a,b)=d is not viable because the And-Or graph for viability has no solution, the implicant algorithm converges to the empty formula:



In the above, a(3), b(4), c(4) and d(5), denote the clause numbers in which unification occurs. We apply the implicant algorithm to the above:

```
                1
               23
              453
            6253  (subsumed by 23)
empty formula
```

The reader should review the definition of viability and then follow the decomposition that occurs in the And-Or graph which simply traces the recursive application of the viability test. The And-Or graph contains no duplicate nodesj hence, in expanding node 4 we cycle back to node 2, creating a cyclic graph. All the circularity which occurs during the viability test will produce cycles in the And-Or graph which are analyzed by the implicant algorithm to determine viability.

We impose the condition that the nodes of a viability graph be mutually distinct so that if a successor node already exists in the graph we arc back to that unique representation in the graph. The leaf nodes of a viability graph will necessarily be either unsolvable argument nodes (the viability test is not satisfied) or solved unification nodes (a term unifies with the argument of an equality literal). The root inequality is viable if and only if the implicant algorithm converges to at least one implicant composed purely of solved unification nodes. In RUE resolution in strong form we are required to resolve to the topmost viable disagreement set (each inequality must be viable and nearest the topmost disagreement of the complementary literals resolved.

The And-Or graph for the viability of an inequality will always be finite since the number of distinct terms or subterms in S is finite and duplicate nodes are not permitted in the graph.

We will present the strict formalization of the concept of an And-Or graph applied to viability, defining the different types of nodes present in the graph, namely, inequality nodes, argument nodes, unification nodes, function-matching nodes and disagreement set nodes, in a proximate future paper.

References

[Lu and Subrahmanian, 1988] James J. Lu and V. S. Subrahmanian. Completeness Issues in RUE-NRF Deduction. Logic Programming LPRG-TR-88-24 Technical Report, Syracuse University, 1988.

[Digricoli and Harrison, I986] Vincent J. Digricoli and Malcolm C. Harrison. Equality-based Binary Resolution. Journal of the ACM, 33(2):253-289, April 1986.

[Chang and Slagle, 1971] C. Chang and James Slagle. An Admissable and Optimal Algorithm for Searching And/Or Graphs. Artificial Intelligence 2:117-128, 1971.

[Nilsson, 1971] Nils J. Nilsson. Problem Solving Methods in Artificial Intelligence. McGraw-Hill, 1971.

[Robinson, I965] J.A. Robinson. A Machine Oriented Logic Based on the Resolution Principle. Journal of ACM, 12(1):23-41, January 1965.