PATH RESOLUTION WITH LINK DELETION

Neil V. Murray and *Erik Rosenthal*

State University of NY at Albany
Department of Computer Science
Albany, NY 12222

University of New Haven
Department of Mathematics
West Haven, CT 06516

ABSTRACT

We introduce a graphical representation of quantifier-free predicate calculus formulas and a new rule of inference which employs this representation. The new rule is an amalgamation of resolution and Prawitz analysis which we call path resolution. Path resolution allows Prawitz analysis of an arbitrary subgraph of the graph representing a formula. If such a subgraph is not large enough to demonstrate a contradiction, a path resolvent of the subgraph may be generated with respect to the entire graph. This generalizes the notions of large inference present iu hyper-resolution, clash-resolution, NC-resolution, and PL-resolution.

Two forms of path resolution are described for which deletion of the links resolved upon preserves the spanning property.

1. Introduction

Since about 1960 most of the effort in automated deduction has been concerned with refutation systems. The initial emphasis was on Prawitz analysis [7,9,10,12,20], in which the unsatisfiability of a sentence is deduced without inferring new formulas. By 1965, with the advent of resolution and (later) paramod illation, the emphasis shifted almost completely toward the use of inference [15,21,22,23,27]. Most of the work done within both schools of thought employed conjunctive or disjunctive normal form. More recently there have been adaptations of both techniques toward the use of unnormalized or less-normalized formulas [2,11 ,16,17,19,25,2ft].

We introduce a new rule of inference, path resolution, which operates on a graphical representation of quantifier-free predicate calculus formulas. The new rule is an amalgamation of resolution and Prawitz analysis. Our goal in the design of path resolution is to retain some of the advantages of both Prawitz analysis and resolution methods, and yet to avoid to some extent their disadvantages.

The main advantage of Prawitz analysis is that, except for variants of original formulas, no new formulas are inferred which rapidly expand the search space. However, except for adding variants, Prawitz analysis is an all or nothing time-bound search for a contradiction. In contrast, resolution and other inference based methods store the progress made at each inference by retaining the inferred formula. Eventually localized evidence of a contradiction is produced (usually the empty clause). The required multiple variants of formulas are automatically (and often excessively) generated. But each new formula introduced interacts with others, expanding the search rapidly in both time *and space..*

One of the disadvantages of resolution is its reliance on conjunctive normal form. We avoid conjunctive and disjunctive normal form and the duplication of literals that their use may necessitate, since path resolution operates on formulas in negation normal form. We have found that the analysis is greatly simplified by a representation of NNF formulas which we call *semantic graphs.'*

Path resolution allows Prawitz analysis of an arbitrary subgraph of the existing graphical representation of formulas. If such a subgraph is not large enough to demonstrate a contradiction, a path resolvent of the subgraph may be generated with respect to the entire graph. Path resolution operations include (properly) all resolution-based inferences of which the authors are aware, such as hyper-resolution, clash-resolution, NC-resolution, and UL-resolut ion.

It will be obvious to the informed reader that the work reported here, while new, has been synthesized from a number of important ideas developed by others. The most influential of these are Robinson's clash resolution [21], Kowalski s connection-graph procedure [14], the work of Andrews [2] and Bibel [3,4] on paths, matrices, and the spanning property, and the non-clausal systems of Stickel [25] and of Waldinger and Manna [2ft].

In the next section we introduce the notation and terminology required for expressing formulas and their semantics in terms of semantic graphs and paths. We further develop this formalism in section 3; section 4 introduces the rule of inference, path resolution. Section 5 contains a sample refutation, and section ft introduces two link deletion strategies. We omit many proofs for lack of space; they are available in [18].

2. Semantic Graphs and Paths

A *semantic graph* is a means of representing a logical formula, and *paths* determine the semantics of the graph. We assume that the reader is familiar with the definitions of *atom, literal, formula, resolution,* and *unification.* We will consider only quantifier-free formulas in which all negations are at the atomic level.

A *semantic graph* is empty, a single node, or a triple *(N,C,D) of nodes, c-arcs,* and *d-arcs,* respectively, where a node is a literal occurrence, a c-arc is a conjunction of two non-empty semantic graphs, and d-arc is a disjunction of two non-empty semantic graphs. Please note that a node is a literal occurrence, so that if a literal occurs twice in a formula, we will label both nodes with that literal. Each semantic graph used in the construction of a semantic graph will be called an *explicit subgraph,* and we shall insist t hat each proper explicit subgraph be contained in exactly one arc. We will use the notation $(G,H)_c$ for the c-arc from *G* to H and similarly use $(G,H)_d$ for a d-arc. The subscript may be omitted if there is no possibility of confusion, and we will use the term graph only for semantic graphs.

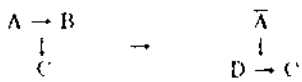
We will consider an empty graph to be an empty disjunction, which is a contradiction A construction of a graph may be thought of as a sequence of c-arcs and d-arcs. There will always be exactly one arc *(X,Y)* with the property that every

other arc is an arc in X or in Y. We call this arc the *final arc* of the graph, and X and Y are the *final subgraphs*. Since this arc completely determines G, we frequently write $G = (X, Y)$.

As an example, the formula

$$((A \wedge B) \vee C) \wedge (\sim A \vee (D \wedge C))$$

is the graph



Note that horizontal arrows are c-arcs, and vertical arrows are d-arcs. There are two d-arcs: (A —> B,C) and (A, D -> C); and there are three c-arcs: (A,B), (D,c), and the entire graph. Some of the explicit subgraphs are each of the nodes, A —> b, and the right-hand part of the graph.

The formulas we are considering are in *negation normal form* (nnf) in that all negations are at the atomic level, and the only connectives used are AND and OR.

If A and B are nodes in a graph, and if a = (X',y') is an arc (c- or d-) with A in X and B in >', we will say that a is the arc *connecting A* and B. If a is a c-arc, we will say that A and B are *c-connected,* and if a is a d-arc, we will say that A and B are *d-connected.*

Lemma 1. Let G he a semantic graph, and let A and B be nodes in G. Then there is a unique arc connecting A and B.

One of the keys to our analysis is the notion of *path.* Let G be a semantic graph. A *partial c-path through G* is a set c of nodes such that any two are connected by a c-arc. We allow the empty set or a singleton set to be a partial c-path. A *c-path* is a partial c-path which is not properly contained in any partial c-path. Notice that any partial c-path is extendible to a c-path, and hence partial c-paths are always subpaths of c-paths. We similarly define d-path using d-arcs instead of e-arcs. The next lemma is an immediate consequence of Lemma 1.

Lemma 2. If A and B are nodes in a graph G, then there is a c-path or a d-path (but not both) containing A and B.

Using paths, we can define the conjunctive normal form of a graph as the conjunction of the d-paths, and the disjunctive normal form as the disjunction of the c-paths. If a formula is multiplied out using the distributive laws, then the graph changes and so may the normalized forms.

Paths can be defined in a somewhat more structural manner. Define a *structural c-path* (scp) c in a graph G as follows: if G consists of a single node A, then c is the 9et {A}; if the final arc of G is a d-arc, then an scp in G is an scp in one of the final subgraphs; and if the final arc is $(X,Y')_c$, then c is the union of an scp in X and an scp in Y. Lemma 3 states that the two formulations of path are equivalent, and as a result we will abandon the terminology structural c-path after the lemma.

Lemma 3. Let G be a semantic graph and let c be a set of nodes in G. Then c is a c-path iff c is an scp.

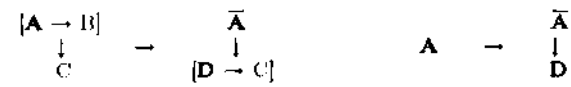
There is an obvious similar statement about d-paths; we leave the proofs to the reader.

The proofs of Lemmas 1 and 3 and of many lemmas and theorems that follow often use induction on the number of arcs in the graph. The base case is always trivial since the graph will then consist of a single node or arc. Since each explicit subgraph will always have fewer arcs than the entire graph, the inductive hypothesis amounts to assuming that the result holds for all explicit subgraphs. As a result, in proofs we do include in this paper, we will ignore the base case and begin by assuming that the result holds for all explicit subgraphs.

We will use the notation c = xy when two pat lis in subgraphs are put together to form a path in a graph as in the lemma.

Lemma 4. Let G be a semantic graph. Then an interpretation 1 satisfies (falsifies) G iff I satisfies (falsifies) every literal on some c-path (d-path) through G.

We will frequently find it useful to consider subgraphs which are not explicit; that is, given any set of nodes, we would like to define that part of the graph which consists of exactly that set of nodes. The previous example is shown below on the left. The subgraph relative to the set {A, C, D} is the graph on the right.



If A is the node set of a graph G, and if N' ' is contained in N, we define G_N , *the subgraph of G relative to* A ' as follows: If A ' = N', then G_N < = G . If the final arc of G is (X',Y), and if the nodes in N ' all appear in A'or in Y, then G_N < = X_N < or G_N =Y_N , respectively. Otherwise, G_N = (X_N > , Y_N >), where this arc is of the same type as (A', Y). The following lemma says in essence that the subgraphs we have defined are the objects we want.

Lemma 5. Let G be a semantic graph with node set A', and let A ' be a subset of A. Then

i) Every node in N ' is a node in some arc in G_N .

ii) If p ' is a path through G_N , then p ' is the restriction to A ' of a path p (of the same type) through G; in particular, if p ' is a partial path in G consisting of nodes from A ' , then p ' is a partial path in G_N .

3. Blocks

Consider the graph (A -+ B) — C. The non-explicit subgraph B —> C "feels" much like an explicit one, and certainly the graph A —► (B -> C) is essentially identical to the original graph. Indeed, the graph A —► C is an explicit subgraph of another essentially identical graph: (A — C) — B. Subgraphs with this property can be characterized with the notion of *block. A c-block C* is a subgraph of a semantic graph with the property that any c-path which includes at least one node from C must pass through C; that is, the subset of the c-path consisting of the nodes which are in C is a c-path through C. *A d-block* is similarly defined with d-paths, and a *full block* is a subgraph which is both a c-block and a d-block. From Lemma 4 we know that the c-paths through a graph determine the semantics of the graph, and that the d-paths also determine the semantics. This might lead one to believe that c-blocks and d-blocks are full blocks. This is not the case, as the following simple example illustrates:

The subgraph relative to {A,B} is obviously a c-block, but it is not a d-block since {AC} is a d-path which meets the subgraph but does not pass through it.

We define *a strong c-block* in a semantic graph G to be a subgraph C of G with the property that every c-path through G contains a c-path through C. *A strong d-block* is similarly

defined.

Lemma 6. If C is a c-block in a semantic graph G, if the final arc (X, Y) of G is a c-arc, and if C meets both X and Y, then C is a strong c-block. Moreover, the subgraphs C_X and C_Y, consisting of the intersections of C with X and Y, are themselves strong c-blocks.

Lemma 7. The union of any number of d-paths in a semantic graph G is a strong c-block. Conversely, a d-path through a strong c-block is a d-path through the entire graph.

Lemma 8. If C is a c-block in G, and if $d_1, d_2, \ldots, d_k$ are d-paths through C, then their union is a c-block in G.

It is obvious that explicit subgraphs are full blocks. There is a sense in which the converse is true. Define two graphs (N, C, D) and (N', C', D') to be *isomorphic* if there exists a bijection $f: N \rightarrow N'$ such that for each A in N, A=f(A), and which preserves c- and d-paths. (By A=f(A), we mean that A and f(A) are the same literal. It is not at all clear what one would even mean by saying that A and f(A) are the same node since they appear in different graphs.) The condition A=f(A) is an essential part of the definition; otherwise, the graphs $A \rightarrow B$ and $\overline{C} \rightarrow C$ would be isomorphic. Notice that being isomorphic is obviously an equivalence relation. A simple example of isomorphic graphs is $A \rightarrow (B \rightarrow C)$ and $(A \rightarrow B) \rightarrow C$ with the "identity" map.

One method of obtaining an isomorphic image of a graph is through commutivity and associativity. Commutivity in a graph amounts to reversing the order in which an arc is formed; e.g., $(Y,X)_i$ instead of $(X,Y)_i$. Associativity in graph amounts to changing the order in which two arcs of the same type are formed; e.g., $(X,(Y,Z)_i)_i$ instead of $((X,Y)_i,Z)_i$. It is immediate that two graphs which are identical except for one such commutation or one such reassociation are isomorphic. Since being isomorphic is transitive, any number of commutations and reassociations will result in an isomorphic graph. In fact, the converse is true. This, and the fact that full blocks are essentially explicit subgraphs, are corollaries of Theorem 1 below.

We define *level* in a graph G as follows: The level of G is 0 and the level of its final arc is 1. Given an arc $\mathbf{a} = (X,Y)$ of level k, if the final arc of X is of the same type as $\mathbf{a}$ then that final arc has level k; otherwise it has level k+1. In either case the level of X is k.

Theorem 1. Let G be a semantic graph, and let B be a full block in G. Then B is a union of level 1 subgraphs of an explicit subgraph of G.

Proof. By induction on the number of arcs in G. Let the final arc of G be (X,Y) and assume that (X,Y) is a c-arc; the case for a d-arc is similar. If B is a subgraph of X or of Y, the induction hypothesis applies, and the theorem holds.

So suppose some nodes of B are in X and some are in Y; recall that B_X and B_Y are the subgraphs of B consisting of the nodes of B which are in X and of those which are in Y, respectively. We show first that B_X and B_Y are themselves full blocks. It is obvious that each is a d-block since the arc (X,Y) is a c-arc: any d-path through G is a d-path through X or a d-path through Y. To see that B_X is a c-block in X, note that B is a strong c-block by Lemma 6. Hence by Lemma 7, B is a union of d-paths through G. Thus B_X is a union of those d-paths which lie in X; i.e., B_X is a strong c-block in X and in particular a full block in X. Similarly, B_Y is a full block and strong c-block in Y.

Now, if the final arc of X is a d-arc, then B_X meets both final subgraphs of X and hence is a strong d-block in X. Since it is both a strong c-block and a strong d-block it must be all of X.

If the final arc of X is a c-arc, let U be any level 1 subgraph of X which meets B_X. The proof will be complete for B_X (and by symmetry for B_Y and hence for all of B) if we show that $U \subset B_X$. But this is clear since the final arc of U must be a d-arc; i.e., the proof is identical to the case when the final arc of X is a d-arc.

Corollaries 1, 2, and 3 are immediate consequences of the theorem. The first corollary (and its generalization, Corollary 4) and the theorem are especially important. Because of the chronological order in which they were originally proven, we will refer to Theorem 1 as the ***second isomorphism theorem*** and to Corollaries 1 and 4 as the ***first isomorphism theorem.***

Corollary 1. Let G be a semantic graph, and let H be a full block in G. Then there is a semantic graph G' and an isomorphism $f: G \rightarrow G'$ such that $f(H)$ is an explicit subgraph of G'.

Corollary 2. If G and H are isomorphic semantic graphs, then H can be formed by reassociating and commuting some of the arcs in G.

Corollary 3. The intersection of two full blocks is a full block.

Corollary 4. Given a semantic graph G and a collection of mutually disjoint full blocks, there is a graph isomorphic to G in which each full block is an explicit subgraph. Moreover, given any two of the blocks, each node in one is c-connected to each node in the other or each node in one is d-connected to each node in the other.

We will find it useful to determine the smallest full block containing a given subgraph; to do that we need the notions of *c- and d- extension.* The c-extension and the d-extension of the entire graph G is G itself. Given a level k subgraph X in G, if $(X,Y)_i$ is an arc in G, then the c-extension of X is the (unique) level k-1 subgraph containing X, and the d-extension is X. The obvious dual applies for $(X,Y)_d$. Given an arbitrary subgraph H of G, to find the smallest full block containing H, let k be the smallest integer such that more than one level k subgraph of G meets H. (Note that k must exist unless H consists of a single node, in which case H is a full block.) Then the smallest full block containing H is the subgraph consisting of those level k subgraphs which meet H.

It will be useful to divide an arbitrary subgraph H of a graph G into c-blocks. We define a c-block C contained in a subgraph H to be *maximal* if no superset of C in H is a c-block. The c-blocks in the definition are assumed to be c-blocks in G, which trivially implies that they are c-blocks in H. But note that if H is not a full block, a subgraph which is a c-block in H might not be a c-block in G. In particular, H may not be a c-block and typically will not be. We define a *proper c-family* of H to be a collection of maximal c-blocks whose union is H. The members of a proper c-family are not in general disjoint.

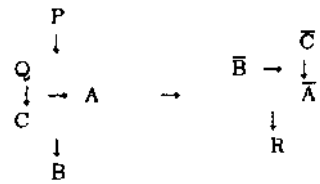
The next theorem says in essence that a certain rule of inference is sound. This may not be entirely obvious from the statement, which is purely structural. First, define the *auxiliary subgraph* $\mathrm{Aux}(H,G)$ of a subgraph H in a semantic graph G to be the subgraph of G relative to the set of all nodes in G which lie on extensions of d-paths through H to d-paths through G. The proof of the next lemma is immediate.

Lemma 9. If H is a subgraph of G, then $\mathrm{Aux}(H,G)$ is empty iff H is a strong c-block. Moreover, $\mathrm{Aux}(H,G)$ cannot contain a d-path through G; if H is a c-block, then so is $\mathrm{Aux}(H,G)$.

Theorem 2. Let $\{C_1, C_2, \ldots, C_k\}$ be a proper c-family in a subgraph H of a graph G. Let I be an interpretation which satisfies a c-path p through G, and suppose that p does not pass through H. Then I satisfies $\mathrm{Aux}(C_i, G)$ for some i.

Proof. Since p does not pass through *H,* it must fail to pass through C, for some i. Hence, p must entirely miss G, , and so p must meet and hence pass through Aux(C, ,G). •

Example 1:



Let *S* be the subgraph relative to {A, B, G, A, B, C}. *S* may be partitioned into the following four c-blocks: {C, B}, {A}, {B}, and {C, A}. However these are not maximal; in particular, the last two c-blocks may be combined to form {C, B}, {A}, and {B, C, A}. But we still do not have a proper c-family. The c-block {A} is not maximal because the node B can be added and {A, B} is still a c-block; {C, B}, {A, B}, and {S, C, A} do form a proper c-family.

Under the conditions of Theorem 2, it is immediate that the disjunction of the auxiliary subgraphs is satisfied by I. (In the above example, any c-path that misses the first c-block must hit subgraph {Q, P}. A c-path missing the second c-block hits {P}, and one missing the third c-block hits {R}.) We denote this disjunction by *P[H,G).* We shall soon see that there are redundancies built into *P(H,G)* which can be eliminated. These redundancies arise from the fact that the auxiliary subgraphs need not be disjoint. Theorem 3 indicates how to remove them. First, define a subgraph of a graph *G* to be a *d-full block* if it is a full block and a strong d-block. Note that for // to be a d-full block in G, *H* must be a disjunction of one or more level 1 subgraphs of *G.*

Theorem 3. Let $\{C_1,,..., C_k\}$ be a proper c-family in a subgraph H of a semantic graph *G,* and let *K* be the intersection of Aux(G, ,G) and Aux(C_j ,*G*). Then *K* is empty or a d-full block in Aux(C,,G).

The significance of Theorem 3 is that the redundancies which appear in *P[H,G)* are its level 1 subgraphs which appear more than once. In example 1, the auxiliary subgraphs of the intersecting c-blocks are the subgraphs relative to {P, Q} and {P}. Their intersection is of course the node P which forms a level 1 subgraph in both auxiliary subgraphs and in their disjunction. However, we can form the disjunction of auxiliary subgraphs to build *P(H,G)* and leave out the redundancies; i.e., use the intersections of the auxiliary subgraphs only once. We denote the resulting graph by *Q(H,G).* We prove that Q(H,G) is unique by defining an object WS(H,G), the *weak split graph of H in* G, and showing that WS(H,G) — *Q(H,G).* (We introduce *strong* split graph in section 6.) We define WS(H,G) as follows:

$$WS(\emptyset, G) = G.$$

$$WS(G, G) = \emptyset.$$

$$WS(H, G) = WS(H_X, X) \vee WS(H_Y, Y) \text{ if } G = (X, Y)_d.$$

$$WS(H, G) = WS(H_X, X) \vee WS(H_Y, Y) \text{ if } G = (X, Y)_c \text{ and } H \text{ meets both } X \text{ and } Y.$$

$$WS(H, G) = WS(H_X, X) \text{ (or } WS(H_Y, Y)) \text{ if } G = (X, Y)_c \text{ and } H \text{ is contained in } X \text{ (in } Y, \text{ respectively)}$$

We will write $WS(H,X)$ for $WS(H_X,X)$. Notice that no nodes appear more than once in $WS(H,G)$, and that $WS(H,G)$ is uniquely defined. In Theorem 4, we show that $Q(H,G) = WS(H,G)$, which tells us that $Q(H,G)$ can be computed without first discovering a proper c-family.

Theorem 4. If H is a subgraph of a semantic graph G, then $Q(H,G) = WS(H,G)$.

It should be noted that only structural redundancies are eliminated by weak split; it is certainly possible that the literals which occur in G will result in $WS(H,G)$ being a tautology. In view of Theorem 4, we will use only the notation $WS(H,G)$. The following corollaries are immediate.

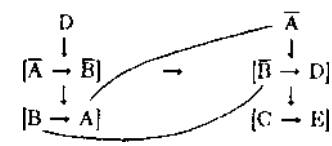
Corollary 1. Let H be a subgraph of a graph G, and let I be an interpretation which satisfies a c-path p through G. Then if p does not pass through H, I satisfies $WS(H,G)$.

Corollary 2. If H is a c-block in a graph G, then $WS(H,G)$ is isomorphic to $Aux(H,G)$.

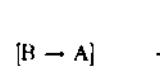
4. Path resolution.

The discussion of paths and graphs gives a (we think useful) representation of logical formulas as graphs. The primary, concern of this paper is *path resolution,* a rule of inference. To this end, we define a *chain* in a graph to be a set of pairs of c-connected nodes such that each pair can simultaneously be made complementary by an appropriate substitution. A *link* is an element of a chain, and a chain is *full* if it is not properly contained in any other chain. A graph *G* is *spanned* by the chain K if every c-path through *G* contains a link from K. Notice that if a graph is spanned by a chain, the graph must be a contradiction since no c-path which contains complementary nodes can be satisfied. If K is a chain, we use the notation G_K for the subgraph of 6' relative to the set of nodes which appear in K. If G_K is spanned by K, K is said to be a *resolution chain,* and G_K is said to be a *resolution subgraph.*

Example 2:



The curves between A and A and between B and B represent links which form a chain. The subgraph relative to the chain



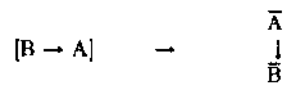
The c-paths arc {B A A) and {B A B}. Each obviously contains a link, so we have a resolution chain and a resolution subgraph.

A rule of inference is of course a procedure which produces a formula from a given formula, and such a procedure is sound if any interpretation which satisfies the original formula also satisfies the inferred formula. Resolution is such a rule; in essence, it applies to formulas in cnf and operates on chains consisting of a single link. We will define a generalization of resolution which we call *path resolution.* It applies to any semantic graph and operates on arbitrary spanning chains.

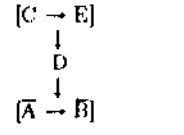
Let K be a resolution chain in a semantic graph G, and let R — G_K . If I is any interpretation which satisfies G, it satisfies a c-path through G. Since *R* cannot be satisfied, c cannot pass through *R.* Hence, by Theorem 4, / *must satisfy WS(R,G).* We call WS(R,G) the *path resolvent* of *R* in G, and we have

Theorem 5. Path resolution is a sound rule of inference.

Consider Example 2 again. We have already seen that



is a resolution subgraph. The c-blocks are [B → A] and the d-path {Ā, B̄}. The auxiliary subgraph of the second is [C → E] since the (partial) d-path {Ā D} does not pass through the resolution subgraph. The path resolvent is



Once a resolvent has been inferred, it may appear natural to conjoin it with G. There is a better way to handle this: the split subgraph should be formed in the smallest full block M containing the resolution chain R, and the resolvent should then be conjoined to M rather than to all of G. That is, we replace M in G with $(M, \mathrm{WS}(M_R, M))_c$. We will call the resulting graph the *weak resultant of G with respect to R* and denote it $\mathrm{WRes}(G,R)$. In the previous examples, (and quite often in general,) M is a union of level 1 c-connected subgraphs of G, and therefore $\mathrm{WRes}(G,R) = (G, \mathrm{WS}(R, G))_c$.

5. A Refutation Proof

Consider the following formulas taken from Chang and Lee [5].

(1) $\forall x\,(E(x) \wedge \sim V(x) \rightarrow \exists y\,(S(x,y) \wedge C(y)))$

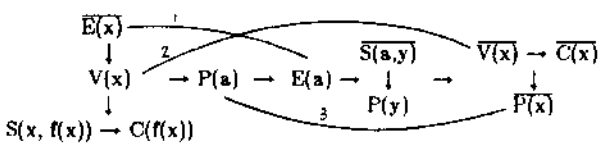
(2) $\exists x\,(P(x) \wedge E(x) \wedge \forall y\,(S(x,y) \Rightarrow P(y)))$

(3) $\forall x\,(P(x) \Rightarrow \sim V(x))$

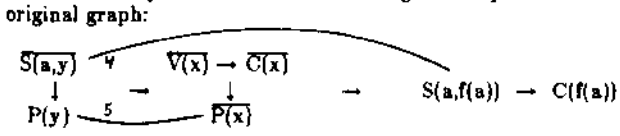
We wish to show that (1), (2), and (3) imply (4) below.

(4) $\exists x\,(P(x) \wedge C(x))$

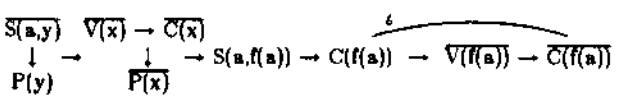
The semantic graph for (1), (2), (3), and the denial of (4) is shown below. Notice that the node $\overline{P(x)}$ is derived from the atom P(x) in both (3) and the denial of (4). If we had represented (3) and ~(4) as separate c-connected graphs, the proof below would still go through: links 3 and 5 would simply contain different occurrences of $\overline{P(x)}$.



Links 1, 2, and 3 span their associated subgraph and therefore form a resolution chain. The weak split of this chain with respect to the entire graph is the graph S(a,f(a)) → C(f(a)). We show this path resolvent below along with a portion of the original graph:



Resolving on links 4 and 5 yields $\overline{V(f(a))} \rightarrow \overline{C(f(a))}$.



It is now easy to see that the link {C(f(a)), C(f(a))} is a resolution chain that spans not only its subgraph but the entire graph. Therefore this link produces the empty graph upon activation. The careful reader may have noticed that this last link is inherited from {C(x), C(f(a))} which could have been added to the resolution chain used in the second inference. The resulting chain would have produced a contradiction shortening our proof to two steps. A binary resolution refutation for these formulas given in [5] uses eight steps.

We do not claim that the search space for determining an appropriate resolution chain is small. However, the existence of a 2-step derivation for this example is certainly somewhat favorable. Moreover, we note that both path resolution steps admit link deletion under certain extensions of Theorems 7 and 8. Space limitations preclude discussion of those extensions here. A derivation of a logic program by NC-resolution is compared to a derivation of the same program by path resolution in [18].

Stickel [25] has pointed out that in a non-clausal connection graph system it may be wise to avoid inferences on non-atomic complementary subformulas. Two reasons cited are that complementary subformulas may be difficult to detect, and that such inferences may be duplicated by resolving on single literals only. Path resolution may help to solve both of these problems. We only link atoms and their complements, not non-atomic structures. The presence of complementary subformulas is detected by a resolution chain. The subformula relative to a resolution chain is frequently (but not always!) a conjunction of complementary subformulas comprised of the literals appearing in the chain, even though these literals may be scattered throughout the entire sentence.

6. Link Deletion

Path resolution is so general that it admits as special cases all resolution-based inference rules (e.g. hyper-resolution, clash resolution UL-resolution, NC-resolution) of which the authors are aware. In fact, if enough copies of formulas from an unsatisfiable set are represented, then a resolution chain will exist whose path resolvent is the empty d-path. We may therefore view semantic graphs and path resolution as a unifying framework for all resolution-based inference and Prawits analysis.

This generality is elegant from a theoretical point of view, but it also admits a proof-search space larger even than that of unrestricted binary resolution. It is natural to ask whether restrictive strategies exist that would take advantage of path resolution's generality, and not just mimic known strategies applicable to (say) clausal logic. One natural restriction is to large chains; that is to avoid resolution chains that are proper sub-chains of currently known resolution chains. Our intuition is that such a strategy is favorable for refutations, and this is an area of ongoing investigation.

Another way to reduce the search space is to delete links after activation. In [3] and [4], for example, Bibel dealt with these issues within binary resolution. We develop two link deletion strategies for path resolution below.

6.1. Link deletion using full blocks

It is more or less the case that a necessary and sufficient condition for a link deletion strategy to be acceptable is that the spanning property be preserved. That is, if a graph is spanned by a set of links, and if a rule of inference which deletes links is applied, then the resulting graph should still be spanned. Theorems 7 and 8 introduce classes of resolution chains for which path resolution has this property. Theorem 6 gives a condition when certain links can be deleted. It should

be pointed out that since we are considering link deletion, we will not in general be dealing with the full set of links, and we will not need to inherit all links. If L is the current set of links in a graph G, and if R is the resolvent with respect to a resolution subgraph of the form $(U, V)_i$ such that the conditions of Theorem 7 or 8 are satisfied, then we need not inherit any links from U or from V to R. The reason is that the only paths in the new graph we need be concerned about are those which pass through $(U, V)_i$; in Theorems 7 and 8 we essentially show that extensions of such paths through R must contain an inherited link, and this allows deletion of the links in R.

Recall that any full block U is a conjunction or a disjunction of level 1 subgraphs of some explicit subgraph H. If the final arc of H is a conjunction, then the c-extension of U is H and the d-extension of U is U itself. The situation is reversed if the final arc of H is a d-arc. We will use the notation $CE(U)$ and $DE(U)$ for the c- and d-extensions, respectively, of U.

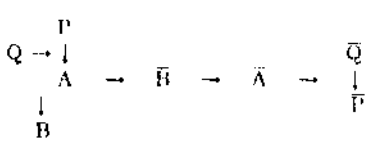
Lemma 10. Let U be a full block in a graph G whose final arc is a c-arc. Let $Z_1 = CE(U)$ and $U_0 = U$, and for $i \geq 1$ define Z_i, Y_i, and U_i recursively as follows: $Y_i = DE(Z_i)$, $Z_i = CE(Y_{i-1})$, and $U_i = Y_i - Z_i$. Then

i) Either $Z_i = G$ or there exists an m such that $Z_i = Y_i = G$ iff $i \geq m$.

ii) If p is a c-path through G, then there exists a k such that p passes through U_k but completely misses U_i for $i < k$. (We do not exclude the possibility that p passes through U_i i.e., that $k = 0$.)

Proof. The proof is actually fairly trivial. Observe that each Z_i is a level 1 subgraph of Y_i, and that each Y_i is a level 1 subgraph of Z_{i+1}. Observe further that U_i is the full block consisting of the disjunction of the level 1 subgraphs of Y_i different from Z_i. To see that i) holds, note that if Z_i is not all of G, then Y_{i-1} is the level 1 subgraph of G containing U. To see that ii) holds, note that there is at least k such that p passes through Y_k. Then p misses Z_i for $i \leq k$. •

Example 3:



Let U_0 be the (subgraph relative to) the single node A which is obviously a full block. Then, as defined in Lemma 10, $Z_1 = \{A\}$, $Y_1 = DE(\{A\}) = \{P, A\}$, $U_1 = Y_1 - Z_1 = \{P\}$, $Z_2 = CE(Y_1) = \{Q, P, A\}$, $Y_2 = DE(Z_2) = \{Q, P, A, B\}$, $U_2 = Y_2 - Z_2 = \{B\}$, and $Z_3 =$ the entire graph. Notice that any c-path must pass through U_0, U_1, or U_2.

Theorem 6 is a generalization of Bibel's *Pure Lemma* [3].

Theorem 6. Let U be a full block in a semantic graph G, let L be a set of links, and suppose that no node in U is contained in a link from L. Let L' be the set of links from L which do not contain a node from $DE(U)$. Then G is spanned by L iff G is spanned by L'.

Proof. Since L' is a subset of L, we need only consider the case when G is spanned by L. If $DE(U) = U$, then L' = L, and there is nothing to prove. If not, let $D = DE(U)$ and $U' = D - U$; let p be a c-path through G. We must show that p contains a link from L'. If p does not pass through U', there is nothing to prove since the links deleted from L all contain nodes from U'. If p does pass through U', let $p = r's$, where r' is a path through U' and s is the rest of p. Let r be any c-path through U. Note that every node in s is c-connected to all of D since D is a full block and the nodes of s are c-connected to some nodes in D (namely, those in r'). Then rs is a c-path through G and must contain a link from L. This link is in L' since no link in L contains a node from U which contains r. This completes the proof since this link must be in s and therefore in p. •

The next theorem gives a condition under which links may be deleted. It says that if $(U, V)_i$ is a resolution subgraph in which U and V are each full blocks in G, and if the links in the chain all go from U to V, then we may delete all links from U to V.

Theorem 7. Let R be a resolution chain in semantic graph G such that G_R has the form $(U, V)_i$, where U and V are full blocks in G, and where each link in R goes from U to V. Suppose further that G is spanned by a set of links L. Let $H = WRes(R, G)$, and let L' be the set of inherited links together with all links in L which do not go from U to V. Then L' spans H.

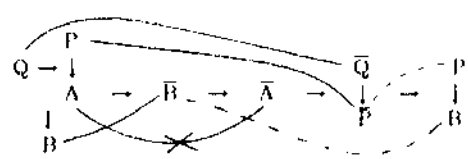
Proof. We proceed by induction on the number of arcs in G. The case for one arc is trivial. If $(X, Y)_d$ is the final arc of G, then R must be entirely contained in X or in Y and the induction hypothesis gives the desired result. If the final arc is $(X, Y)_i$, and if R is contained in X or in Y, again the induction hypothesis applies. So suppose that the final arc is a c-arc and R meets both X and Y. By the first isomorphism theorem, we may assume that U and V are explicit subgraphs. Hence neither can meet both X and Y. Assume that U is contained in X and V is contained in Y. Observe that $H = (G, WS(G_R, G))_i$.

Let p be a c-path through H. We must show that p contains a link from L'. Note that p is the union of c-paths $p_1p_2p_3$ through X, Y, and $S = WS(G_R, G)$, respectively. The only case we need consider is when p passes through G_R and all links in p_1p_2 come from R. Since S is the disjunction of $WS(U, X)$ and $WS(U, Y)$, we assume without loss of generality that p_1 is a c-path through $WS(U, X)$. Since U is an explicit subgraph, we know that its split is isomorphic to $Aux(U, X)$ by Corollary 2 of Theorem 4. Let p_3' be the isomorphic image of p_3 in $Aux(U, X)$.

Using the notation of Lemma 10, apply part ii) of that lemma to p_3'. Since p_3' passes through $Aux(U, X)$, it misses U and hence $k > 0$. As in the proof of Lemma 10, let $p_1 = sr$, where r is a c-path through Z_k. It is obvious that sp_3' is a path through X, so $sp_3'p_2$ contains a link from L. This link is not in R. Moreover, p_2 misses $Aux(V, Y)$, so this link will be inherited, and the proof is complete. •

It is worth noting that all single link chains satisfy the conditions of Theorem 7, and that the proof for this case is no easier since a pair of linked nodes plays the same role structurally as the two full blocks in the theorem.

Consider activating the single-link chain $\{A, \bar{A}\}$ in example 3. This chain clearly satisfies the conditions of Theorem 7. Notice that the resultant graph is still spanned even though the activated link is deleted:



8.2. Another form of path resolution

Consider the soundness argument for path resolution as defined in section 4. In some cases the c-paths that miss the resolution chain are larger than the c-paths in the (weak) split

subgraph. To account for this we define the *strong split* of an arbitrary subgraph H in G, $SS(H,G)$, to be the same as $WS(H, G)$ with the exception that if $G = (X,Y)_c$ and H is contained in X, then $SS(H,G) = SS(H,X) \wedge Y$. If R is a resolution chain, let M be the smallest full block containing R. It is straightforward to verify that if an interpretation satisfies M then it satisfies $SS(G_R, M)$, and hence we may replace M in G with $(M, SS(G_R, M))_c$. We call the resulting graph the *strong resultant of G with respect to R*, and denote it $SRes(G, R)$. The following lemma yields both soundness for strong splits and spanning preservation with link deletion for the class of resolution chains described in Theorem 8.

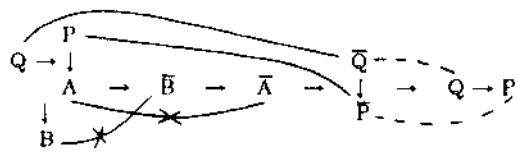
Lemma 11. If X is a c-block in G, then $SS(X, G)$ is isomorphic to the subgraph of G consisting of all nodes that lie on c-paths which miss X.

Proof. Suppose first that $G = (U,V)_d$. Then $SS(X,G) = (SS(X,U), SS(X,V))_d$ and the result follows from the induction hypothesis. If $G = (U,V)_c$, and if X meets both U and V then X is a strong c-block and $SS(X, G)$ is empty. Finally if X is contained in U, then $SS(X,G) = (SS(X,U), V)_c$ and the induction hypothesis applies. •

Theorem 8. Let R be a resolution chain in semantic graph G such that G_R has the form $(X,Y)_c$, where X and Y are c-blocks in G, and where each link in R goes from X to Y. Let G be spanned by a set of links L, and let $M = (U,V)_c$ be the smallest full block containing R, where U and V are chosen to contain X and Y respectively. Then $H = SRes(G, R)$ is spanned by L - R and the links inherited from L.

Proof. Let p be a c-path through H, and let $M' = (M, SS(G_R, M))_c$. If p does not pass through M', then p misses M' entirely since M' is a full block, so p must have a link from L. Assume p passes through M'. Let $p = p_1p_2p_3$ where p_1,p_2,p_3 are c-paths through U, V, and $SS(G_R, M)$ respectively. Since X is a c-block in U, Lemma 9 applies, so let p_3' be the isomorphic image of p_3 in U. Then the c-path $p_3' p_2$ has a link in L that is not in R and that is therefore inherited. •

Consider the two-link chain $\{A, \bar{A}\}$, $\{B, \bar{B}\}$, again from example 3. This chain satisfies the conditions of Theorem 8, but not those of Theorem 7 because $\{A, B\}$ is a c-block but not a full block. Shown below is the outcome of activating this chain with strong split and deleting both links. Of course the graph is still spanned. But notice that had we used weak split, the resolvent would have included only P, and the c-path $\{B, \bar{B}, \bar{A}, \bar{Q}, P\}$ would then be without a link.



7. Summary

We have introduced a graphical representation of formulas and a new rule of inference that employs this representation. The inference rule is path resolution, which is a generalisation of most previous forms of resolution. We have demonstrated the rule's soundness. Completeness is immediate in the absence of the usual connection graph link deletion strategy; resolving exclusively on single-link chains amounts to atomic NC-resolution on unf formulas. We have introduced two classes of chains for which the activation and subsequent deletion of links is shown to preserve the spanning property. In light of Bibel's results [3,4] such link deletion strategies may reasonably be conjectured complete.

References

1. Andrews, P.B. Refutations by mating*. IEEE Transactions on Computers 25,8 (Aug. 1976), 801-807
2. Andrews, P.B. Theorem proving via general matings JACM 28,2 (April 1981), 193-214.
3. Bibel, W. On matrices with connections. JACM 28,4 (Oct 1981). 633-645.
4. Bibel, W. A Strong Completeness Result for the Connection Graph Proof Procedure. Technical Report ATP-3-IV-80.
5. Chang, C.L. and Lee, R.C.T. *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1978.
6. Chang, C.L., and Slagle, JR. Using rewriting rules for connection graphs to prove theorems. Artificial Intelligence 12 (Aug 1979), 159-178.
7. Chinlund, T.J., Davis, M., Hineman. P.G., and McIlroy, M.D. Theorem proving by matching Bell Laboratory, 1964.
8. Clark, K. The Synthesis and Verification of Logic Programs. Third Conference "on Automated Deduction, August 1977
9. Davis, M. and Putnam, H. A computing procedure for quantification theory. JACM. vol. 7 (1960), 201-215.
10. Davis, M. Eliminating the irrelevant from mechanical proofs. Proe. Symp. of Applied Mathematics 15 (1963), 15-30.
11. de Champeaux, D. Sub-problem finder and instance checker - two cooperating processors for theorem provers Proc. 4th Workshop on Automated Deduction, Austin. Texas, Feb 1979, 110-114.
12. Gilmore, P.C, A Proof method for quantification theory. IBM Journal of Research and Development, vol. 4 (1960). 28-35
13. Henschen, L.G. Theorem proving by covering expressions J.ACM, 26,3 (July 1979). 385-400
14. Kowalski, R. A proof procedure using connection graphs J.ACM 22,4 (Oct. 1975), 572-595.
15. McCharen, J., Overbeek, R. and Wos, L. Problems and experiments for and with automated theorem-proving programs. IEEE Transactions on Computers, C-25,8 (Aug. 1976). 773-782
16. Murray, N.V. Completely non-clausal theorem proving Artificial Intelligence 18.1 (Jan. 1982), 67-85.
17. Murray, N.V. An experimental theorem prover using fast unification and vertical path graphs. Fourth National Conf. of Canadian Society of Computational Studies of Intelligence, U. of Saskatchewan, May 1982.
18. Murray, N.V. and Rosenthal, E. Semantic graphs. Technical Report 84-12, Department of Computer Science, SUNT at Albany, Nov. 1984.
19. Nilsson, N.J. A production system for automatic deduction Technical Note 148, SRI International, 1977.
20. Prawitz, D. An improved proof procedure Theoria 26 (I960). 102-139.
21. Robinson, G.A. and Wos, L. Paramodulation and theorem proving in first order theories with equality *Machine Intelligence 4,* 1969, Edinburgh University Press.
22. Robinson, J.A. A machine oriented logic based on the resolution principle. J.ACM 12,1 (1965), 23-41.
23. Robinson, J.A. Automatic deduction with hyper-resolution International Journal of Computer Mathematics, 1 (1965), 227-234.
24. Robinson, J.A. "Theoretical Approaches to Non-Numerical Problem Solving," Springer-Verlag, New York, Inc., 1970, 2-20
25. Stickel, M.L. A nondausal connection-graph resolution theorem-proving program. Proc. AAAI-82 Nat Conf on Artificial Intelligence, Pittsburgh, Pennsylvania. Aug 1982, 229-233.
26. Waldinger, R. and Manna, Z. A deductive approach to program synthesis. ACM TOPLAS 2,1 (1980), 90-121

Wos, L., Carson, D. and Robinson, G. Efficiency and completeness of the set of support strategy in theorem proving J.ACM 12,4 (1965), 536-541.