-

From rameshag Sun Apr 17 13:23:28 1994 Received: from karp.cs.albany.edu (karp.cs.albany.edu [128.204.2.32]) by cs.albany.edu (8.6.8/HUB01) with ESMTP id NAA18148 for <nvm@cook>; Sun, 17 Apr 1994 13:23:23 -0400 From: Anavai G Ramesh <rameshag> Received: (rameshag@localhost) by karp.cs.albany.edu (8.6.8/CLI) id NAA07565 for nvm; Sun, 17 Apr 1994 13:23:17 -0400 Message-Id: <199404171723.NAA07565@karp.cs.albany.edu> Subject: LPAR camera To: nvm (Neil Murray) Date: Sun, 17 Apr 1994 13:23:15 -0400 (EDT) X-Mailer: ELM [version 2.4 PL17] Content-Type: text Content-Length: 59824 Status: RO

On Anti-Links [†]

Bernhard Beckert[‡], *Reiner Hähnle*[‡], *Anavai Ramesh*[*], *and Neil V. Murray*[*]

[‡] Institute for Logic, Complexity and Deduction Systems, Universität Karlsruhe, Karlsruhe, Germany. (email: beckert / haehnle@ira.uka.de)

[*] Institute for Programming & Logics, Department of Computer Science, University at Albany, Albany, NY 12222. (email: rameshag / nvm @cs.albany.edu)

Abstract.

The concept of *anti-link* is defined, and useful equivalence-preserving operations on propositional formulas based on anti-links are introduced. These operations eliminate a potentially large number of subsumed paths in a negation normal form formula. The operations have linear time complexity in the size of that part of the formula containing the anti-link.

These operations are useful for prime implicant/implicate algorithms because most of the computational effort in such algorithms is spent on subsumption checks.

1. Introduction

Many algorithms have been proposed to compute the prime implicates of propositional boolean formula. Most algorithms [1,2,3,4,14] assume that the input is either in conjunctive normal form (CNF) or in disjunctive normal form (DNF). Other algorithms [10] require the input to be a conjunction of DNF formulas. In [12], a set of techniques for finding the prime implicates of formulas in negation normal form (NNF) is proposed. Those techniques are based on *dissolution*, an inference rule introduced in [8], and on an algorithm called PI. Classes of formulas have been discovered for which these techniques are polynomial but for which any CNF/DNF-based technique must be exponential in the size of the input. Ngair has also introduced similar examples; however, our method is more general than Ngair's which is based on order theory [10].

In [12] the PI algorithm is described; there, PI is used to enumerate all the prime implicates of a *full dissolvent*, an NNF formula that has no conjunctive links (defined later). PI repeatedly does subsumption checks to keep intermediate results as small as possible. However these checks are expensive. Most result in failure, and they have to be done on sets which can be exponentially large. The time required for these operations can be reduced by using a more compact representation of the intermediate results [1], but avoiding as many such checks as possible is the focus of this paper.

We show that the full dissolvent can be restructured before applying PI such that many non-prime implicates are removed without doing subsumption checks at

[†] This research was supported in part by National Science Foundation Grant CCR-9101208

all. We define *disjunctive* and *conjunctive anti-links*[1] in NNF formulas, and we identify operations to remove such anti-links and their associated subsumed paths. This leaves fewer subsumption checks for the PI algorithm.

In the next section we describe our path semantics viewpoint and our graphical representation of formulas in classical logic. In Section 3 we introduce *anti-links* and develop useful equivalence-preserving operations based on them.

2. Foundations: Facts on Formulas in Negation Normal Form

We assume the reader to be familiar with the notions of *atom, literal*, and *formula* from classical logic. We consider only formulas in *negation normal form* (NNF): The only connectives used are conjunction and disjunction, and all negations are at the atomic level. This restriction is reasonable, since formulas that contain implications, equivalences, and negations at any level can be converted to NNF in polynomial time. We deal only with propositional logic in this paper, although some of the following results like the Path Dissolution Rule are completely general.

In this section, we introduce a number of technical terms and definitions that are treated in detail in [9]. They are required for the development of the anti-link operations defined in Section 3, and they make the paper self-contained even for readers not familiar with dissolution.

2.1. Semantic Graphs

A *semantic graph G* is 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 semantic graphs, and a d-arc is a disjunction of two semantic graphs. Any of N,C,D may be empty. If N is empty, G is either *true* (empty conjunction) or *false* (empty disjunction). Each semantic graph used in the construction of a semantic graph is called an *explicit subgraph*, and each proper explicit subgraph is contained in exactly one arc. Note that when a graph contains occurrences of *true* and *false*, the obvious truth-functional reductions apply. Unless otherwise stated, we will assume that semantic graphs are automatically so reduced and that empty graphs are *false*. We will typically use *G* to refer to both the graph and to the corresponding node set when the meaning is evident from context.

We use the notation $(X,Y)_c$ for the c-arc from *X* to *Y* and similarly use $(X,Y)_d$ for a d-arc; the subscript may be omitted when no confusion is possible. Arbitrary subformulas are denoted by upper case italic letters; plain upper case letters are used for single nodes.

In Figure 1, the formula on the left is displayed graphically on the right. Note that c-arcs and d-arcs are indicated by the usual symbols for conjunction and disjunction. Essentially, the only difference between a semantic graph and a formula in NNF is the point of view, and we will use either term depending upon the desired emphasis. For a more detailed exposition, see [9].

(Ramesh and Murray) and by Deutsche Forschungsgemeinschaft within the Schwerpunktprogramm Deduktion (Hähnle and Beckert).

[1] Anti-links and some associated operators were first proposed by Beckert and Hähnle – personal communication. The first motivation for studying anti-links arose in connection with regular clausal tableau calculi (Letz, p. 114 [6]). The anti-link rule as it will be defined later can be viewed as an implementation of the regularity condition in [6] for the propositional non-clausal case (Letz considered the first-order clausal case). There, refinements of general inference rules are considered, whereas the anti-link rule allows implementation as a preprocessing step.

$$((\neg C \wedge A) \vee D \vee E) \wedge (\neg A \vee (B \wedge C)) \quad \equiv$$

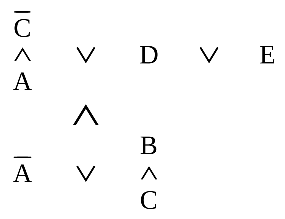


Figure 1.

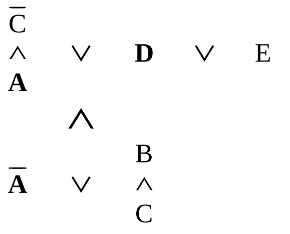
If A and B are nodes in a graph, and if $\mathbf{a} = (X,Y)_\alpha$ is an arc ($\alpha = \mathrm{c}$ or $\alpha = \mathrm{d}$) with A in *X* and B in *Y*, we say that **a** is the arc *connecting* A and B, and that A and B are α-*connected.* In Figure 1, C is c-connected to each of B, A, $\bar{C}$, D, and E and is d-connected to $\bar{A}$.

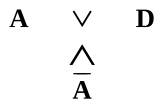
Let *G* be a semantic graph. A *partial c-path through G* is a set of nodes such that any two are c-connected, and a *c-path* through *G* is a partial c-path that is not properly contained in any partial c-path. The c-paths of the graph in Figure 1 above are: {$\bar{C}$, A, $\bar{A}$}, {$\bar{C}$, A, B, C}, {D, $\bar{A}$}, {D, B, C}, {E, $\bar{A}$}, {E, B, C}. We similarly define d-path using d-arcs instead of c-arcs. The following lemma is obvious.

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

2.1.1. Subgraphs

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





For a precise definition of subgraph, see [9].

2.1.2. Blocks

The most important subgraphs are the *blocks*. A *c-block H* is a subgraph of a semantic graph with the property that any c-path p that includes at least one node from *H must pass through H*; that is, the subset of p consisting of the nodes that are in *H* must be a c-path through *H*. A *d-block* is similarly defined with d-paths. In Figure 1, the subgraph relative to the node set {A,D,E,$\bar{A}$,C} is a c-block. However, it is not a d-block since the d-path {$\bar{A}$,B} restricted to the subgraph is {$\bar{A}$}, which is a proper sub-path of {$\bar{A}$,C} in the subgraph.

A *full block* is a subgraph that is both a c-block and a d-block. One way to envision a full block is to consider conjunction and disjunction as n-ary connectives. Then a full block is a subset of the arguments of one connective, i.e., of one explicit subformula. For example, in Figure 1, {$\bar{C}$,A,E} is a full block. Full blocks may be treated as essentially explicit subgraphs, and the Isomorphism Theorem from [7] assures us that they are the only structures that may be so treated. For example,

$\{\overline{C}, A, E\}$ can be written as $\begin{smallmatrix}\overline{C}\\ \wedge\\ A\end{smallmatrix} \vee E$ or as $(\{\overline{C}, A\}, E)_d$.

Let *H* be a full block; *H* is a conjunction or a disjunction of fundamental subgraphs of some explicit subgraph *M*. If the final arc (main connective) of *M* is a conjunction, then we define the *c-extension* of *H* to be *M* and the *d-extension* of *H* to be *H* itself. The situation is reversed if the final arc (main connective) of *M* is a d-arc. We will use the notation CE(*H*) and DE(*H*) for the c- and d-extensions, respectively, of *H*. In Figure 1, CE($\overline{A}$) = $\overline{A}$ and DE($\overline{A}$) = $\overline{A} \vee \begin{smallmatrix}B\\ \wedge\\ C\end{smallmatrix}$.

In this paper, we compute c- and d-extensions of single nodes only. Single nodes are always full blocks and so testing for this property will be unnecessary. If we assume that formulas are represented as n-ary trees, computing these extensions can be done in constant time; we merely determine whether the given node's parent is a conjunction or a disjunction, and the appropriate extension is then either the node itself or the parent.

2.2. Path Dissolution

A *c-link* is defined to be a complementary pair of c-connected nodes; d-connected complementary nodes form a d-link. Unless stated otherwise, we use the term link to refer to a c-link. Path dissolution is in general applicable to collections of links; here we restrict attention to single links. Suppose then that we have literal occurrences A and $\overline{A}$ residing in conjoined subgraphs *X* and *Y*, respectively. Consider, for example, the link $\{A, \overline{A}\}$ in Figure 1. Then the entire graph *G* = (*X* ∧ *Y*) is the smallest full block containing the link, where

$$X \;=\; \begin{smallmatrix}\overline{C}\\ \wedge\\ A\end{smallmatrix} \vee D \vee E \qquad \text{and} \qquad Y \;=\; \overline{A} \vee \begin{smallmatrix}B\\ \wedge\\ C\end{smallmatrix} \quad .$$

The *c-path complement* of an arbitrary subgraph *H* with respect to *X*, written CC(*H*, *X*), is defined to be the subgraph of *X* consisting of all literals in *X* that lie on paths that do not contain nodes from *H*; the *c-path extension* of *H* with respect to *X*, written CPE(*H*, *X*), is the subgraph containing all literals in *X* that lie on paths that *pass through H*.

In Figure 1, CC(A, *X*) = (D ∨ E); CPE(A, *X*) = ($\overline{C}$ ∧ A). (Note that CPE has two arguments whereas CE has but one; intuitively, CE has an implicit second argument that is always the entire graph in which the explicit argument occurs. For instance, CPE(A, *G*) = ($\overline{C}$ ∧ A ∧ *Y*) and CE(A) = CPE(A,*X*) = ($\overline{C}$ ∧ A).)

It is intuitively clear that the paths through (*X* ∧ *Y*) that do not contain the link are those through (CPE(A, *X*) ∧ CC($\overline{A}$, *Y*)) plus those through (CC(A, *X*) ∧ CPE($\overline{A}$,*Y*)) plus those through (CC(A, *X*) ∧ CC($\overline{A}$, *Y*)). The reader is referred to [9] for the formal definitions of CC and of CPE and for the proofs of the lemmas below. (The CC and CPE definitions are also presented in Section 3.3).

Lemma 2. Let *H* be an arbitrary subgraph of *G*. The c-paths of CPE(*H*, *G*) are precisely the c-paths of *G* that pass through *H*.

Corollary. CPE(*H*, *G*) is exactly the subgraph of *G* relative to the set of nodes that lie on c-paths that pass through *H*.

Lemma 3. Let *H* be an arbitrary subgraph of *G*. The c-paths of CC(*H*, *G*) are precisely the c-paths of *G* that do not pass through *H*.

Corollary. CC(*H*, *G*) is exactly the subgraph of *G* relative to the set of nodes that lie on c-paths that do not pass through *H*.

Lemma 4. If H is a c-block, then CC(H, G) $\cup$ CPE(H, G) = G.

In the development of anti-link operations, we will require the dual operations of CC and CPE. We use DC for the *d-path complement* and DPE for the *d-path extension* operators. Their definitions and properties are straightforward by duality, and the above lemmas and corollaries about CC and CPE all hold in dual form for DC and DPE. The above informal treatment of these operators is adequate for an intuitive description of dissolution. However, precise definitions are given in Section 3, where they will be required in proving the correctness of the anti-link operations introduced there.

Let $H = \{A, \overline{A}\}$ be a link, and let $M = (X, Y)_c$ be the smallest full block containing H. The only way that H can be a single c-block is if H is a full block (it is trivially a d-block). In that case, $H = M$, and A and $\overline{A}$ must be (up to commutations and reassociations) arguments of the same conjunction.

In general we define DV(H, M), the *dissolvent of H in M*, as follows: If H is a single c-block, then DV(H, M) = CC(A, M) = CC($\overline{A}$, M) = *false*. Otherwise (i.e., if H consists of two c-blocks),

$$\mathrm{DV}(H,\ M) \;=\; \begin{matrix}\mathrm{CPE}(A,\ X)\\ \wedge\\ \mathrm{CC}(\overline{A},\ Y)\end{matrix} \;\vee\; \begin{matrix}\mathrm{CC}(A,\ X)\\ \wedge\\ \mathrm{CPE}(\overline{A},\ Y)\end{matrix} \;\vee\; \begin{matrix}\mathrm{CC}(A,\ X)\\ \wedge\\ \mathrm{CC}(\overline{A},\ Y)\end{matrix}$$

It follows from the corollaries and Lemma 4 that either of the two graphs shown below may also be used for DV(H, M).

$$\begin{matrix}X\\ \wedge\\ \mathrm{CC}(\overline{A}, Y)\end{matrix} \;\vee\; \begin{matrix}\mathrm{CC}(A, X)\\ \wedge\\ \mathrm{CPE}(\overline{A}, Y)\end{matrix} \qquad \text{or} \qquad \begin{matrix}\mathrm{CC}(A, X)\\ \wedge\\ Y\end{matrix} \;\vee\; \begin{matrix}\mathrm{CPE}(A, X)\\ \wedge\\ \mathrm{CC}(\overline{A}, Y)\end{matrix}$$

The three versions of DV(H, M) are not identical as graphs, but all three do have the identical c-paths: all those of the original full block M except those of CPE(A, X) $\wedge$ CPE($\overline{A}$, Y), i.e., except those through the link.

Theorem 1. Let H be a link in a semantic graph G, and let M be the smallest full block containing H. Then M and DV(H, M) are logically equivalent.

A proof of Theorem 1 (in its full generality, where H is an arbitrary dissolution chain) can be found in [9].

We may therefore select an arbitrary link H in G and replace the smallest full block containing H by its dissolvent, producing (in the ground case) an equivalent graph. We call the resulting graph the *dissolvent of G with respect to H* and denote it Diss(G,H). Since the paths of the new graph are all that appeared in G except those that contained the link, this graph has strictly fewer c-paths than the old one. As a result, finitely many dissolutions (bounded above by the number of c-paths in the original graph) will yield a linkless equivalent graph. This proves

Theorem 2. At the ground level, path dissolution is a strongly complete rule of inference.

2.3. Prime Implicates/Implicants

We briefly summarize basic definitions regarding implicates. The treatment for implicants is completely dual and is indicated by appropriate dual expressions in parentheses.

-

A disjunction (conjunction) P of literals is an implicate (implicant) of a formula G, iff $G \models$ P (P $\models G$).

A disjunction (conjunction) D subsumes another D′ iff D $\models$ D′ (D′ $\models$ D).

If a disjunction (conjunction) D′ is not equivalent to *true* (*false*) then D subsumes D′ iff D $\subseteq$ D′. *True* (*false*) is subsumed by all disjunctions (conjunctions). A *true* disjunction (*false* conjunction) can subsume another *true* disjunction (*false* conjunction) only.

A disjunction (conjunction) D is a prime implicate (implicant) of a formula G iff

1) D is not *true* (*false*).
2) D is an implicate (implicant) of G.
3) For all literals l_i in D, $G \not\models (D - \{l_i\})$ $((D - \{l_i\}) \not\models G)$.

2.4. Fully Dissolved Formulas

If we dissolve in a semantic graph G until it is linkless, we call the resulting graph the *full dissolvent of* G and denote it by FD(G). Observe that FD(G) is dependent on the order in which links are activated. However, the set of c-paths in FD(G) is unique: It is exactly the set of satisfiable c-paths in G. Because FD(G) is link-free, the consequences, i.e., implicates, of G are represented in the d-paths of FD(G). In a dual manner, we may define dissolution for disjunctive links; in that case, FD(G) has no disjunctive links, and the implicants of G are represented in the c-paths of FD(G). These relationships are made precise by Theorem 3 below.

In the discussion that follows, we will often refer to subsumption of d- and c-paths rather than of disjuncts and conjuncts. Paths are defined as sets of literal occurrences, but with regard to subsumption, we consider the *literal sets* of paths. We denote by l(p) the literal set of path p. In this way, no change in the standard definitions is necessary. The theorem below was proved in [12].

Theorem 3. In any non-empty formula in which no c-path (d-path) contains a link, every implicate (implicant) of the formula is subsumed by some d-path (c-path) in the formula.

Corollary: Every prime implicate (implicant) of a reduced DNF (CNF) formula, i.e., one with no *false* conjuncts (*true* disjuncts), is subsumed by some d-path (c-path) in the formula.

This follows directly from Theorem 3 because such a DNF (CNF) formula has no c-paths (d-paths) with links.

In [12], the prime implicates of G are computed by first obtaining FD(G); then, knowing that all implicates are present in the d-paths of FD(G), the PI algorithm computes the set of prime implicants $\psi(\mathrm{FD}(G))$, where

$$\begin{aligned}\psi(\mathcal{F}) \;=\; \{P \mid\; & (P \text{ is a d-path through } \mathcal{F}) \wedge \\ & (P \neq \textit{true}\,) \wedge (\forall \text{ d-paths Q through } \mathcal{F},\; l(Q) \not\subset l(P))\} \;.\end{aligned}$$

When used in this way, PI extracts all unsubsumed (non-tautological) d-paths from an NNF formula without c-links. In general, PI computes $\psi(\mathcal{F})$ for an arbitrary NNF formula $\mathcal{F}$.

3. Subsumed Paths and Anti-Links

Our goal in this section is to first identify as many subsumed paths as possible in an efficient manner and then eliminate them. The presence of anti-links (both disjunctive and conjunctive) in a graph may indicate that subsumed d-paths are present in the graph. We now define anti-links and then discuss ways to identify and remove subsumed paths due to anti-links.

Let M=$(X,Y)_d$ be a d-arc in a semantic graph *G* and let A_X and A_Y be occurrences of the literal A in X and in Y respectively. Then we call $\{A_X,A_Y\}$ a *disjunctive anti-link*. Note that M is the smallest full block containing the anti-link. If M=$(X,Y)_c$ is a c-arc in a semantic graph *G* and if A_X and A_Y are nodes in X and in Y respectively, then we call $\{A_X,A_Y\}$ a *conjunctive anti-link*.

3.1. Redundant Anti-links

We now identify those disjunctive anti-links which do imply the presence of subsumed paths. We say a disjunctive anti-link $\{A_X,A_Y\}$ with respect to the graph *G* is *redundant* if either $CE(A_X) \neq A$ or $CE(A_Y) \neq A$.

Let $\{A_X,A_Y\}$ be a disjunctive anti-link in graph *G*, where $M = (X,Y)_d$ is the smallest full block containing the anti-link. We define $\mathcal{DP}_{A_X,A_Y,G}$ to be the set of all d-paths of *M* which pass through both $CE(A_X) - \{A_X\}$ and A_Y or through both $CE(A_Y) - \{A_Y\}$ and A_X. Consider the following example:

$$\begin{array}{ccccccc} A & \vee & C & & A & & \\ & \wedge & & \vee & \wedge & & \\ & B & & & E & \vee & C \end{array}$$

The two occurrences of A form a disjunctive anti-link; *M* is the entire graph, and *X* and *Y* are the left and right arguments of the main disjunction, respectively. Because $CE(A_Y) - \{A\} = Y - \{A\} = (E \vee C)$ and $DPE(A_X, X) = A \vee C$, $\mathcal{DP}_{A_X,A_Y,G}$ contains the d-path $p = \{A_X, C, E, C\}$. But since $CE(A_X) = A_X$, there are no paths through $CE(A_X) - \{A_X\}$; p is the only member of $\mathcal{DP}_{A_X,A_Y,G}$. Nevertheless, the anti-link is redundant, and p is subsumed by $p' = \{A_X, C, A_Y\}$ (with literal set {A,C}). Notice that had *M* been embedded in a larger graph *G′*, every d-path q containing p in *G′* is subsumed by a corresponding d-path q′ that differs from q only in that q′ contains p′ instead of p.

In general, one or both of the literals in a redundant anti-link $\{L_X, L_Y\}$ is an argument of a conjunction, and $\mathcal{DP}_{L_X,L_Y,G} \neq \emptyset$. In the above example, the two occurrences of C are both arguments of disjunctions, and thus comprise a non-redundant anti-link for which $\mathcal{DP}_{C_X,C_Y,G} = \emptyset$.

Although only redundant disjunctive anti-links contribute directly to subsumed d-paths, non-redundant anti-links do not prohibit the existence of subsumed paths. However, such non-redundant anti-links do not themselves provide any evidence that such paths are in fact present.

Theorem 5. Let $\{A_X,A_Y\}$ be a redundant disjunctive anti-link in semantic graph *G*. Then each d-path in $\mathcal{DP}_{A_X,A_Y,G}$ is properly subsumed by a d-path in *G* that contains the anti-link.

Proof: Recall that a d-path (c-path) in a graph *G* is said to *pass through* a subgraph *X* of *G* if the path when restricted to the set of nodes in *X*, forms a d-path (c-path) in *X*. Let $p \in \mathcal{DP}_{A_X,A_Y,G}$, and assume without loss of generality that p passes through both $CE(A_X) - \{A_X\}$ and A_Y. Note that $CE(A_X) - \{A_X\}$ is non-empty and that $M = (X \vee Y)$ is the largest full block containing the anti-link. We may write $CE(A_X)$ as $(A_X \wedge C_1 \wedge \cdots \wedge C_n)$, where $n \geq 1$.

Let $p = p_X p_Y p_o$ where p_X and p_Y are p restricted to *X* and to *Y*, respectively, and p_o is p restricted to nodes outside of both *X* and *Y*. By construction, $A_X \notin p_X$ and thus p_X passes through some C_i, $1 \leq i \leq n$. So $p_X = p_X' p_{C_i}$, where p_{C_i} is p_X restricted to C_i, and hence $p = p_X' p_{C_i} p_Y p_o$. The d-path $p_X' \cup \{A\}$ clearly passes through *X*, and since $A_Y \in p_Y$, $p' = p_X' A_X p_Y p_o$ subsumes p. □

3.2. An Anti-Link Operator

The identification of redundant disjunctive anti-links can be done easily by checking to see if either $\mathrm{CE}(A_X) \neq A_X$ or $\mathrm{CE}(A_Y) \neq A_Y$. After identifying a redundant anti-link, it is possible to remove it using the *disjunctive anti-link dissolvent* (DADV) operator; in the process, all d-paths in $\mathcal{DP}_{A_X,A_Y,G}$ are eliminated, and the two occurrences of the anti-link literal are collapsed into one. Let $\{A_X, A_Y\}$ be a disjunctive anti-link and let $M = (X,\ Y)_d$ be the smallest full block containing the anti-link. Then

$$\mathrm{DADV}(\{A_X,\ A_Y\},\ M) \quad = \quad \begin{array}{rcl} \mathrm{DC}(A_X,X) & \vee & \mathrm{DC}(A_Y,Y) \\ & \wedge & \\ \mathrm{DC}(\mathrm{CE}(A_X),X) & \vee & \mathrm{DPE}(A_Y,Y) \\ & \wedge & \\ \mathrm{DPE}(A_X,X) & \vee & \mathrm{CC}(A_Y,Y) \end{array} \ .$$

Consider again the example from Section 3.1:

$$\begin{array}{ccc} \begin{array}{ccc} A & \vee & C \\ & \wedge & \\ & B & \end{array} & \vee & \begin{array}{ccc} & A & \\ & \wedge & \\ E & \vee & C \end{array} \end{array}$$

We have $\mathrm{DC}(A_X,\ X) = B$ and $\mathrm{DC}(A_Y,\ Y) = (E \vee C)$, so the upper conjunct in DADV is $(B \vee E \vee C)$. For the middle conjunct, $\mathrm{CE}(A_X) = A_X$, $\mathrm{DC}(\mathrm{CE}(A_X),\ X) = B$, and $\mathrm{DPE}(A_Y,\ Y) = A_Y$; this conjunct is $(B \vee A)$. Finally in the lower conjunct, $\mathrm{DPE}(A_X,\ X) = (A \vee C)$ and $\mathrm{CC}(A_Y,\ Y) = \emptyset$ (*false*), so this reduces to $(A \vee C)$. The result is:

$$\mathrm{DADV}(\{A_X,\ A_Y\},\ M) \quad = \quad \begin{array}{ccl} B & \vee & E \vee C \\ & \wedge & \\ B & \vee & A \\ & \wedge & \\ A & \vee & C \end{array}$$

We point out that although DADV produces a CNF formula in this simple example, in general it does not. In particular, the above graph can be simplified as the consequence of easily recognizable conditions, and the resulting graph is not in CNF. For the details, see Case 1 of Section 3.5.

3.3. Extension and Path Complement Operators

A number of more primitive operators are used in the definition of DADV; they are described in [9] and have been presented informally in Section 2. We present the formal definitions here in order to prove Lemma 5 below, and, in the next subsection, to verify that DADV has the desired properties.

Let H be an arbitrary subgraph of G. Then

$$\mathrm{CPE}(\emptyset,\ G) \ = \ \emptyset \ (\textit{false}) \qquad \text{and} \qquad \mathrm{CPE}(G,G) \ = \ G$$

$$\mathrm{CPE}(H,\ G) \quad = \quad \bigvee_{i=1}^{n} \mathrm{CPE}(H_{F_i},F_i) \qquad \text{if the final arc of } G \text{ is a d-arc}$$

$$\mathrm{CPE}(H,\ G) \quad = \quad \bigwedge_{i=1}^{k} \mathrm{CPE}(H_{F_i},F_i) \ \wedge \ \bigwedge_{j=k+1}^{n} F_j \qquad \text{if the final arc of } G \text{ is a c-arc}$$

where $F_1, \cdots, F_k$ are the fundamental subgraphs of G that meet H, and $F_{k+1}, \cdots, F_n$ are those that do not.

$$\mathrm{DPE}(\varnothing, G) = \varnothing\ (\mathit{true}) \quad \text{and} \quad \mathrm{DPE}(G, G) = G$$

$$\mathrm{DPE}(H, G) = \bigwedge_{i=1}^{n} \mathrm{DPE}(H_{F_i}, F_i)$$
if the final arc of G is a c-arc

$$\mathrm{DPE}(H, G) = \bigvee_{i=1}^{k} \mathrm{DPE}(H_{F_i}, F_i) \vee \bigvee_{j=k+1}^{n} F_j$$
if the final arc of G is a d-arc

where $F_1, \cdots, F_k$ are the fundamental subgraphs of G that meet H, and $F_{k+1}, \cdots, F_n$ are those that do not.

Using the same notation we define the c- and d-path complements of H in G as follows:

$$\mathrm{CC}(\varnothing, G) = G \quad \text{and} \quad \mathrm{CC}(G, G) = \varnothing\ (\mathit{false})$$

$$\mathrm{CC}(H, G) = \bigvee_{i=1}^{n} \mathrm{CC}(H_{F_i}, F_i)$$
if the final arc of G is a d-arc

$$\mathrm{CC}(H, G) = \bigwedge_{i=1}^{k} \mathrm{CC}(H_{F_i}, F_i) \wedge \bigwedge_{j=k+1}^{n} F_j$$
if the final arc of G is a c-arc

where $F_1, \cdots, F_k$ are the fundamental subgraphs of G that meet H, and $F_{k+1}, \cdots, F_n$ are those that do not.

$$\mathrm{DC}(\varnothing, G) = G \quad \text{and} \quad \mathrm{DC}(G, G) = \varnothing\ (\mathit{true})$$

$$\mathrm{DC}(H, G) = \bigwedge_{i=1}^{n} \mathrm{DC}(H_{F_i}, F_i)$$
if the final arc of G is a c-arc

$$\mathrm{DC}(H, G) = \bigvee_{i=1}^{k} \mathrm{DC}(H_{F_i}, F_i) \vee \bigvee_{j=k+1}^{n} F_j$$
if the final arc of G is a d-arc

where $F_1, \cdots, F_k$ are the fundamental subgraphs of G that meet H, and $F_{k+1}, \cdots, F_n$ are those that do not.

Lemma 5. If G is a graph and A is a literal occurrence in G, then

$$\mathrm{CC}(\mathrm{A}, G) = (\mathrm{DPE}(\mathrm{A}, G) - \{\mathrm{A}\}) \wedge \mathrm{DC}(\mathrm{CE}(\mathrm{A}), G) .$$

Proof: We prove the lemma by showing that the formula on the left and the formula on the right possess exactly the same set of d-paths. The proof is done via induction on the syntactic structure of G.

-

If G is a literal, then $G = \mathrm{A}$ and both CC(A, G) and ((DPE(A, G) − {A}) ∧ DC(CE(A), G)) are empty. (DC(CE(A), G) = DC(A, A) = *true*, but (DPE(A, G) − {A}) = {A} − {A} = *false* = CC(A, A) .)

If G = (X ∨ Y), then without loss of generality assume A belongs to X. Hence CC(A, G) = CC(A, X) ∨ Y. By the induction hypothesis, the d-paths of CC(A, X) are just those of (DPE(A, X) − {A}) ∧ DC(CE(A), X)). So CC(A, G) = (DPE(A, X) − {A}) ∧ DC(CE(A), X)) ∨ Y.

Now consider the right hand side of the equation. Since A is in X, DPE(A, G) = DPE(A, X) ∨ Y. Therefore, DPE(A, G) − {A} = DPE(A, X) − {A} ∨ Y. Also, CE(A) will be disjoint from Y, and thus DC(CE(A), G) = DC(CE(A), X) ∨ Y. Therefore we can write the right hand side of the equation as (DPE(A, X) − {A} ∨ Y) ∧ (DC(CE(A), X) ∨ Y). By factoring out the subgraph Y we get an equivalent subgraph ((DPE(A, X) − {A}) ∧ DC(CE(A), X)) ∨ Y having the same d-paths. But this is just the graph to which the left hand side reduced via the induction hypothesis.

Finally suppose G = (X ∧ Y); again assume that A is in X. Now there are two subcases to consider.

a) If CC(A, G) is the empty graph , then A in X is not d-connected to any other subgraph in X. Hence X is of the form $\mathrm{A} \wedge C_1 \wedge \cdots \wedge C_n$ (where $n \geq 0$). But then $G = \mathrm{A} \wedge C_1 \wedge \cdots \wedge C_n \wedge Y$, CE(A) = G, and DPE(A, G) = DPE(A, X) = A. As a result, DC(CE(A), G) and DPE(A, G) − {A} are both the empty subgraph.

b) If CC(A, G) is not empty, then CC(A, G) = CC(A, X) ∧ Y, and CC(A, X) cannot be the empty graph. Therefore, by the induction hypothesis, CC(A, G) = (DPE(A, X) − {A}) ∧ DC(CE(A), X) ∧ Y. Focusing now on the right hand side of the equation, DPE(A, G) = DPE(A, X) by definition. The c-extension of A can only include nodes from X (otherwise, CC(A, G) would be empty, contrary to the subcase b) condition). Therefore, DC(CE(A), G) = DC(CE(A), X) ∧ Y. Therefore the right hand side of the equation is (DPE(A, X) − {A}) ∧ (DC(CE(A), X) ∧ Y). This is just the result obtained for the left hand side in this subcase. □

3.4. Correctness of DADV

In Theorem 6 below we show that DADV({A_X,A_Y},G) is logically equivalent to G and does not contain those d-paths in $\mathcal{DP}_{\mathrm{A}_X,\mathrm{A}_Y,G}$.

Theorem 6. Let $M = (X \vee Y)$ be the smallest full block containing {A_X,A_Y}, a disjunctive anti-link in semantic graph G. Then DADV({A_X, A_Y}, M) is equivalent to M and differs in d-paths from M as follows: Those d-paths in $\mathcal{DP}_{\mathrm{A}_X,\mathrm{A}_Y,M}$ are not present, and any d-path of M containing the anti-link is replaced by a path with the same literal set having only one occurrence of the anti-link literal.

Proof: Note that A_X and A_Y are literal occurrences (and hence d-blocks) in X and in Y respectively. By the duals of Lemmas 2, 3, and 4, X is equivalent to DC(A_X,X) ∧ *DPE(*A_X,X), and from the distributive law,

$$M \quad = \quad \begin{array}{c} \mathrm{DC}(\mathrm{A}_X, X) \ \vee \ Y \\ \wedge \\ \mathrm{DPE}(\mathrm{A}_X, X) \ \vee \ Y \end{array} \quad .$$

Similarly, Y is equivalent to DC(A_Y,Y) ∧ *DPE(*A_Y,Y), and we expand the upper occurrence of Y and distribute.

$$
M \quad = \quad \begin{array}{rcl}
\mathrm{DC}(\mathrm{A}_X, X) & \vee & \mathrm{DC}(\mathrm{A}_Y, Y) \\
 & \wedge & \\
\mathrm{DC}(\mathrm{A}_X, X) & \vee & \mathrm{DPE}(\mathrm{A}_Y, Y) \\
 & \wedge & \\
\mathrm{DPE}(\mathrm{A}_X, X) & \vee & Y
\end{array}
$$

By the duals of Lemmas 2 and 3, not only have we rewritten M equivalently, but the d-paths of M have been preserved. We will continue to rewrite M; our goal is to eventually put it in an equivalent form in which the paths of $\mathcal{DP}_{\mathrm{A}_X,\mathrm{A}_Y,M}$ have been omitted.

Consider the d-paths of $\mathrm{DC}(\mathrm{A}_X, X)$ – the d-paths in X that miss A_X. They either miss $\mathrm{CE}(\mathrm{A}_X)$, the c-extension of A_X, or pass through $\mathrm{CE}(\mathrm{A}_X) - \{\mathrm{A}_X\}$. Hence we have $\mathrm{DC}(\mathrm{A}_X, X) = \mathrm{DPE}(\,(\mathrm{CE}(\mathrm{A}_X) - \{\mathrm{A}_X\}\,), X) \wedge \mathrm{DC}(\mathrm{CE}(\mathrm{A}_X), X)$, and d-paths are preserved. By replacing the lower occurrence of $\mathrm{DC}(\mathrm{A}_X, X)$ in the previous graph, we get the following graph M$'$ which is equivalent to M and has the same d-paths as M.

$$
M' \quad = \quad \begin{array}{rcl}
\mathrm{DC}(\mathrm{A}_X, X) & \vee & \mathrm{DC}(\mathrm{A}_Y, Y) \\
 & \wedge & \\
\begin{array}{c}
\mathrm{DPE}(\,(\mathrm{CE}(\mathrm{A}_X) - \{\mathrm{A}_X\}\,), X) \\
\wedge \\
\mathrm{DC}(\mathrm{CE}(\mathrm{A}_X), X)
\end{array} & \vee & \mathrm{DPE}(\mathrm{A}_Y, Y) \\
 & \wedge & \\
\mathrm{DPE}(\mathrm{A}_X, X) & \vee & Y
\end{array}
$$

Every d-path in the subgraph $\mathrm{DPE}(\,(\mathrm{CE}(A_X) - \{\mathrm{A}_X\}\,), X) \vee \mathrm{DPE}(\mathrm{A}_Y, Y)$ is in $\mathcal{DP}_{\mathrm{A}_X,\mathrm{A}_Y,M}$. By Theorem 5, all these paths are subsumed by other d-paths. Therefore, we can remove the subgraph $\mathrm{DPE}(\,(\mathrm{CE}(A_X) - \{\mathrm{A}_X\}\,), X)$ from M' while preserving equivalence to get the graph M$''$ shown below.

$$
M'' \quad = \quad \begin{array}{rcl}
\mathrm{DC}(\mathrm{A}_X, X) & \vee & \mathrm{DC}(\mathrm{A}_Y, Y) \\
 & \wedge & \\
\mathrm{DC}(\mathrm{CE}(\mathrm{A}_X), X) & \vee & \mathrm{DPE}(\mathrm{A}_Y, Y) \\
 & \wedge & \\
\mathrm{DPE}(\mathrm{A}_X, X) & \vee & Y
\end{array}
$$

Again by using arguments dual to the one given earlier for X , we have

$$
Y \quad = \quad \begin{array}{c}
\mathrm{DPE}(\mathrm{A}_Y, Y) \\
\wedge \\
\mathrm{DPE}(\,(\mathrm{CE}(\mathrm{A}_Y) - \{\mathrm{A}_Y\}\,), Y) \\
\wedge \\
\mathrm{DC}(\mathrm{CE}(\mathrm{A}_Y), Y)
\end{array}
$$

In particular, the d-paths of the two are identical.

Replacing Y in M$''$ we find that every d-path in the subgraph $\mathrm{DPE}(\mathrm{A}_X, X) \vee \mathrm{DPE}(\,(\mathrm{CE}(A_Y) - \{\mathrm{A}_Y\}\,), Y)$ is in $\mathcal{DP}_{\mathrm{A}_X,\mathrm{A}_Y,M}$. Again by Theorem 5, these paths are also subsumed by other d-paths. Therefore we can remove the subgraph $\mathrm{DPE}(\,(\mathrm{CE}(A_Y) - \{\mathrm{A}_Y\}\,), Y)$ and preserve equivalence; M''' results.

$$M''' \quad = \quad \begin{array}{rcl} \mathrm{DC}(\mathrm{A}_{X,}\ X) & \vee & \mathrm{DC}(\mathrm{A}_Y,\ Y) \\ & \wedge & \\ \mathrm{DC}(\mathrm{CE}(\mathrm{A}_X), X) & \vee & \mathrm{DPE}(\mathrm{A}_Y,\ Y) \\ & \wedge & \\ \mathrm{DPE}(\mathrm{A}_X,\ X) & \vee & \begin{array}{c} \mathrm{DPE}(\mathrm{A}_Y, Y) \\ \wedge \\ \mathrm{DC}(\mathrm{CE}(\mathrm{A}_Y), Y) \end{array} \end{array}$$

The d-paths in M''' are those d-paths of M excluding the d-paths in $\mathcal{DP}_{\mathrm{A}_X,\mathrm{A}_Y,M}$. Consider now the d-paths of $\mathrm{DPE}(\mathrm{A}_X, X) \vee \mathrm{DPE}(\mathrm{A}_Y, Y)$ in M'''. They are exactly those of M (and of M''') that contain the anti-link: They each contain two occurrences of the literal A. Hence we can remove the node A_Y from $\mathrm{DPE}(\mathrm{A}_Y, Y)$ to get M''''.

$$M'''' \quad = \quad \begin{array}{rcl} \mathrm{DC}(\mathrm{A}_{X,}\ X) & \vee & \mathrm{DC}(\mathrm{A}_Y,\ Y) \\ & \wedge & \\ \mathrm{DC}(\mathrm{CE}(\mathrm{A}_X),\ X) & \vee & \mathrm{DPE}(\mathrm{A}_Y,\ Y) \\ & \wedge & \\ \mathrm{DPE}(\mathrm{A}_X,\ X) & \vee & \begin{array}{c} \mathrm{DPE}(\mathrm{A}_Y,\ Y) - \{\mathrm{A}_Y\} \\ \wedge \\ \mathrm{DC}(\mathrm{CE}(\mathrm{A}_Y),\ Y) \end{array} \end{array}$$

Applying Lemma 5 to M'''' we get the following graph which is $\mathrm{DADV}(\{\mathrm{A}_X, \mathrm{A}_Y\}, M)$.

$$\mathrm{DADV}(\{\mathrm{A}_X, \mathrm{A}_Y\},\ M) \quad = \quad \begin{array}{rcl} \mathrm{DC}(\mathrm{A}_{X,}\ X) & \vee & \mathrm{DC}(\mathrm{A}_Y,\ Y) \\ & \wedge & \\ \mathrm{DC}(\mathrm{CE}(\mathrm{A}_X),\ X) & \vee & \mathrm{DPE}(\mathrm{A}_Y,\ Y) \\ & \wedge & \\ \mathrm{DPE}(\mathrm{A}_X,\ X) & \vee & \mathrm{CC}(\mathrm{A}_Y,\ Y) \end{array}$$

In constructing $\mathrm{DADV}(\{\mathrm{A}_X,\mathrm{A}_Y\}, M)$, we have removed only subsumed d-paths and altered only d-paths that contain the anti-link by collapsing the double occurrence of the anti-link literal. Hence $\mathrm{DADV}(\{\mathrm{A}_X,\mathrm{A}_Y\}, M)$ is equivalent to M, does not contain the anti-link, and does not contain any d-path of $\mathcal{DP}_{\mathrm{A}_X,\mathrm{A}_Y,M}$. $\square$

Theorem 6 gives us a method to remove disjunctive anti-links and some subsumed d-paths: Simply identify a redundant anti-link $H = \{\mathrm{A}_X,\mathrm{A}_Y\}$ and the smallest full block M containing it, and then replace M by $\mathrm{DADV}(H, M)$. The cost of this operation is proportional to the size of the graph replacing M. Also, c-connected literals in M do not become d-connected in $\mathrm{DADV}(H, M)$. Thus truly new disjunctive anti-links are not introduced. However, parts of the graph may be duplicated, and this may give rise to additional copies of anti-links not yet removed. Nevertheless, persistent removal of redundant disjunctive anti-links (in which case $\mathcal{DP}_{\mathrm{A}_X,\mathrm{A}_Y,M} \neq \emptyset$) is a terminating process, because the number of d-paths is strictly reduced at each step. This proves

Theorem 7. Finitely many applications of the DADV operation on redundant anti-links will result in a graph without redundant disjunctive anti-links, and termination of this process is independent of the choice of anti-link at each step. $\square$

Although we can remove all the redundant disjunctive anti-links in the graph, this process can introduce new conjunctive anti-links. Such anti-links may indicate the presence of subsumed d-paths, but the situation is not as favorable as with disjunctive anti-links – see Section 3.7.

3.5. Simplifications

Obviously, DADV($\{A_X, A_Y\}$, M) can be syntactically larger than M. Under certain conditions we may use simplified alternative definitions for DADV. These definitions result in formulas which are syntactically smaller than those that result from the general definition. The following is a list of possible simplifications.

1. If CE(A_X) = A_X (and CE(A_X) $\neq X$), then DC(CE(A_X), X) = DC(A_X , X). Therefore by (possibly non atomic) factoring on DC(A_X, X) and observing that (DC(A_Y, Y) $\wedge$ DPE(A_Y, Y)) = Y, DADV($\{A_X, A_Y\}$, M) becomes

$$\begin{array}{ccc} \mathrm{DC}(A_X,X) & \vee & Y \\ & \wedge & \\ \mathrm{DPE}(A_X,X) & \vee & \mathrm{CC}(A_Y,Y) \end{array}$$

It turns out that this rule applies to the example used in Sections 3.1 and 3.2, and shown below.

$$\begin{array}{ccc} \begin{array}{ccc} A & \vee & C \\ & \wedge & \\ & B & \end{array} & \vee & \begin{array}{ccc} & A & \\ & \wedge & \\ E & \vee & C \end{array} \end{array}$$

Since CE(A_X) = A_X, the simplified rule for this case results in the following graph.

$$\begin{array}{ccc} B & \vee & \begin{array}{c} E \vee C \\ \wedge \\ A \end{array} \\ & \wedge & \\ A & \vee & C \end{array}$$

2. If CE(A_X) = X, then DC(CE(A_X), X) = $\emptyset$ (*true*). Hence DPE(A_X, X) = A_X and DC(A_X, X) = $X - \{A_X\}$. DADV($\{A_X, A_Y\}$, M) becomes

$$\begin{array}{ccc} X - \{A_X\} & \vee & \mathrm{DC}(A_Y,Y) \\ & \wedge & \\ A_X & \vee & \mathrm{CC}(A_Y, Y) \end{array}$$

3. If both Case 1 and Case 2 apply, then CE(A_X) = X = A_X, and the above formula simplifies to

$$A_X \vee \mathrm{CC}(A_Y, Y)$$

Note that in all the above versions of DADV, the roles of X and Y can be interchanged.

3.6. Disjunctive Anti-Links and Factoring.

It is interesting to note that the DADV operation contains factoring (i.e., the ordinary application of the distributive law to a pair of conjunctions containing a common argument) as a special case. This is just the condition for Case 2 above except that both $\mathrm{CE}(\mathrm{A}_X) = X$ and $\mathrm{CE}(\mathrm{A}_Y) = Y$ hold. Under these conditions, $\mathrm{DADV}(\{\mathrm{A}_X, \mathrm{A}_Y\}, M)$ becomes

$$\begin{array}{c} X - \{\mathrm{A}_X\} \;\vee\; Y - \{\mathrm{A}_Y\} \\ \wedge \\ \mathrm{A} \end{array}$$

This is the graph obtained by disjunctive factoring [9].

The DADV operator also captures the absorption law (or merging). If A_X and A_Y are both arguments of the same disjunction, then $X = \mathrm{A}_X$, $Y = \mathrm{A}_Y$, and $\mathrm{DADV}(\{\mathrm{A}_X, \mathrm{A}_Y\}, M) = \mathrm{A}_X$. Note, however, that the anti-link is not redundant in this case.

3.7. Conjunctive Anti-Links

There are conjunctive anti-links that always indicate the presence of d-paths that are subsumed by others, and they are easy to detect. However, the conditions to be met are much more restrictive than those for redundant disjunctive anti-links. Consider a conjunctive anti-link $\{\mathrm{A}_X, \mathrm{A}_Y\}$, where the smallest full block M containing the anti-link is $(\mathrm{A}_X \wedge Y)$. Every d-path in Y which passes through A_Y will be subsumed by the d-path consisting of the single literal A_X. Hence we can replace Y by $\mathrm{DC}(\mathrm{A}_Y, Y)$.

This is a kind of dual to Case 3 of the simplified versions of DADV discussed earlier. There, the anti-link $\{\mathrm{A}_X, \mathrm{A}_Y\}$ is disjunctive and $M = (\mathrm{A}_X \vee Y)$. The simplified DADV operation just replaces Y by $\mathrm{CC}(\mathrm{A}_Y, Y)$. Note that the conjunctive anti-link operation above removes subsumed d-paths, whereas the Case 3 disjunctive anti-link operation can either remove paths or merely remove the second occurrence of the anti-link literal on paths that contain the anti-link. Both operations involve d-paths, and both have strictly dual operations that would affect c-paths instead.

4. Conclusion and Future Work

We have introduced anti-links and defined useful equivalence-preserving operations on them. These operations can be employed so as to strictly reduce the number of d-paths in an NNF formula. Unlike path dissolution, which removes unsatisfiable (or tautological, in the dual case) paths, anti-link operations remove subsumed paths without any direct checks for subsumption. This is significant for prime implicate computations, since such computations tend to be dominated by subsumption checks.

Some experimental results on a dissolution- and PI-based prime implicate system are reported in [12]. That system should be extended to include anti-link operations, so as to test their effectiveness in practice. The applicability of our techniques to Ngair's examples [10] is also worthy of study, because for many of his examples the full dissolvent contains useful anti-links. Also, his method requires a normal form that is somewhat more general than CNF or than DNF, whereas our techniques require only NNF and are thus more general.

-

References

1. de Kleer, J. An improved incremental algorithm for computing prime implicants. *Proceedings* of *AAAI-92*, 780-785.
2. Jackson, P., and Pais, J. Computing prime implicants. *Proceedings* of *CADE-10*, Kaiserslautern, W. Germany, July, 1990. In *LNAI*, Springer-Verlag, Vol. 449, 543-557.
3. Jackson, P. Computing prime implicants incrementally. *Proceedings* of *CADE-11*, Saratoga Springs, NY, June, 1992. In *LNAI*, Springer-Verlag, Vol. 607, 253-267.
4. Kean, A., and Tsiknis, G. An incremental method for generating prime implicants/implicates. *Journal of Symbolic Computation* **9** (1990), 185-206.
5. Kean, A., and Tsiknis, G. Assumption based reasoning and clause management systems. *Computational Intelligence* **8**,1 (Nov. 1992),1-24.
6. Letz, R. First-order calculi and proof procedures for automated deduction. Ph.D. thesis, TH Darmstadt, June 1993.
7. Murray, N.V., and Rosenthal, E. Inference with path resolution and semantic graphs. *J.ACM* 34,2 (April 1987), 225-254.
8. Murray, N.V., and Rosenthal, E. Path dissolution: A strongly complete rule of inference. *Proceedings* of *AAAI-87*, Seattle, WA, July 12-17, 1987, 161-166.
9. Murray, N.V., and Rosenthal, E. Dissolution: Making paths vanish. *J.ACM* **40**,3 (July 1993), 504-535.
10. Ngair,T. A new algorithm for incremental prime implicate generation. *Proceedings of IJCAI-93,* Chambery, France, August, 1993.
11. Przymusinski, T.C. An algorithm to compute circumscription. Artificial Intelligence **38** (1989), 49-73.
12. Ramesh, A., and Murray, N.V. Non-clausal deductive techniques for computing prime implicants and prime implicates. *Proceedings* of *LPAR-93.* St. Petersburg, Russia, July 13-20,1993. In *LNAI*, Springer-Verlag, Vol. 698, 277-288.
13. Reiter, R. and de Kleer, J. Foundations of assumption-based truth maintenance systems: preliminary report. *Proceedings of AAAI-87*, Seattle, WA, July 12-17, 1987, 183-188.
14. Slagle, J.R., Chang, C.L., and Lee, R.C.T. A new algorithm for generating prime implicants. *IEEE Transactions on Computers,* **C-19(4)** (1970), 304-310.
15. Strzemecki, T. Polynomial-time algorithms for generation of prime implicants. *Journal of Complexity* **8** (1992), 37-63.