Fast Subsumption Checks Using Anti-Links and Strictly Pure Full Blocks †

TR 94-1

Anavai Ramesh   *Neil V. Murray*

Inst. for Programming & Logics
Dept. of Computer Science
State Univ. of N.Y. at Albany
Albany, NY 12222

rameshag / nvm @cs.albany.edu

Abstract

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

The problem of removing all subsumed paths in an NNF formula is shown to be NP-hard, even though such formulas may be small relative to the size of their path sets. The general problem of determining whether there exists a pair of subsumed paths associated with an arbitrary anti-link is shown to be NP-complete. Additional techniques based on *strictly pure full blocks* **are introduced and are also shown to eliminate redundant subsumption checks. The effectiveness of these techniques is examined with respect to some benchmark examples from the literature.**

† This research was supported in part by National Science Foundation Grant CCR-9101208. This paper appears in a condensed form in the *Proceedings* of the *National Conference on Artificial Intelligence*, Seattle, WA, July 31 - August 4, 1994.

Fast Subsumption Checks Using Anti-Links and Strictly Pure Full Blocks

Anavai Ramesh *Neil V. Murray*

1. Introduction

The consequences of a ground formula, expressed as minimal implied clauses, are useful in certain approaches to non-monotonic reasoning [5,11,15], where all consequences of a formula set (e.g., the support set for a proposed commonsense conclusion) are required. Minimal conjunctions that imply a formula are useful in situations where satisfying models are desired, as in error analysis during hardware verification. Such minimal implied clauses are the formula's *prime implicates*, and the minimal conjunctions that imply it are its *prime implicants*.

Many algorithms have been proposed to compute the prime implicates of propositional boolean formula. Most algorithms [1,2,3,4,16] assume that the input is either in conjunctive normal form (CNF) or in disjunctive normal form (DNF). The algorithm of [10] requires the input to be a conjunction of DNF formulas. In [12] we propose a set of techniques for finding the prime implicates of formulas in negation normal form (NNF). Our techniques are based on *dissolution*, an inference rule introduced in [8], and on an algorithm called PI. We have discovered classes of formulas for which our 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] we describe the PI algorithm; 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. Many 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 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. In Section 4, complexity issues are discussed and some NP-completeness results are proven. Section 5 introduces further techniques based on strictly pure subformulas. The effectiveness of our techniques on certain benchmark formulas described by Ngair [10] is explored.

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 and negations at any level can be converted to NNF in polynomial time.

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

---

[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.

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 $(X,Y)_{\mathrm{c}}$ for the c-arc from X to Y and similarly use $(X,Y)_{\mathrm{d}}$ 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 below, the formula on the left is displayed graphically on the right:

$$((\neg \mathrm{C}\wedge \mathrm{A}) \vee \mathrm{D} \vee \mathrm{E}) \wedge (\neg \mathrm{A} \vee (\mathrm{B}\wedge \mathrm{C})) \qquad \equiv \qquad \begin{array}{ccccc} \overline{\mathrm{C}} & & & & \\ \wedge & \vee & \mathrm{D} & \vee & \mathrm{E} \\ \mathrm{A} & & & & \\ & \wedge & & & \\ & & \mathrm{B} & & \\ \overline{\mathrm{A}} & \vee & \wedge & & \\ & & \mathrm{C} & & \end{array}$$

Figure 1.

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].

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, $\overline{\mathrm{C}}$, D, and E and is d-connected to $\overline{\mathrm{A}}$.

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: $\{\overline{\mathrm{C}}$, A, $\overline{\mathrm{A}}\}$, $\{\overline{\mathrm{C}}$, A, B, C$\}$, $\{$D, $\overline{\mathrm{A}}\}$, $\{$D, B, C$\}$, $\{$E, $\overline{\mathrm{A}}\}$, $\{$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, $\overline{A}$,} is the graph on the right.

$$
\begin{array}{ccccc}
\overline{C} & & & & \\
\wedge & \vee & \mathbf{D} & \vee & E \\
\mathbf{A} & & & & \\
 & \wedge & & & \\
 & & B & & \\
\overline{\mathbf{A}} & \vee & \wedge & & \\
 & & C & &
\end{array}
\qquad\qquad
\begin{array}{ccc}
\mathbf{A} & \vee & \mathbf{D} \\
 & \wedge & \\
 & \overline{\mathbf{A}} &
\end{array}
$$

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,$\overline{A}$,C} is a c-block. However, it is not a d-block since the d-path {$\overline{A}$,B} restricted to the subgraph is {$\overline{A}$}, which is a proper subpath of {$\overline{A}$,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. In Figure 1, {$\overline{C}$,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, the full block {$\overline{C}$, A, E} can be written as as $(\{\overline{C}, A\}, E)_d$; i.e., we can regard the upper part of the graph as $(\{\overline{C}, A, E\}, D)_d$.

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 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 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, $\mathrm{CE}(\overline{\mathrm{A}}) = \overline{\mathrm{A}}$ and $\mathrm{DE}(\overline{\mathrm{A}}) = \overline{\mathrm{A}} \vee \begin{array}{c}\mathrm{B}\\ \wedge \\ \mathrm{C}\end{array}$.

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 $\overline{\mathrm{A}}$ residing in conjoined subgraphs X and Y, respectively. Consider, for example, the link $\{\mathrm{A}, \overline{\mathrm{A}}\}$ in Figure 1. Then the entire graph $G = (X \wedge Y)$ is the smallest full block containing the link, where

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

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*. (Because we consider only single link dissolution, the first arguments of CC and CPE will be simple literals when these operations are used in the construction of dissolvents in the examples below. However, this is not the case in Section 3, and hence the above descriptions of these operators are in full generality.)

In Figure 1, CC(A, X) = (D $\vee$ E); CPE(A, X) = ($\overline{\mathrm{C}}$ $\wedge$ 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) = ($\overline{\mathrm{C}}$ $\wedge$ A $\wedge$ Y) and CE(A) = CPE(A,X) = ($\overline{\mathrm{C}}$ $\wedge$ A).)

It is intuitively clear that the paths through ($X \wedge Y$) that do not contain the link are those through (CPE(A, X) $\wedge$ CC($\overline{\mathrm{A}}$, Y)) plus those through (CC(A, X) $\wedge$ CPE($\overline{\mathrm{A}}$,Y)) plus those through (CC(A, X) $\wedge$ CC($\overline{\mathrm{A}}$, Y)). If we dissolve on the {A, $\overline{\mathrm{A}}$} link in Figure 1 (using the compact form of dissolution shown on the right just prior to Theorem 1 below), the graph that results is:

$$\begin{array}{ccccc} & & & & \mathrm{C}\\ & & & & \wedge\\ \mathrm{D} & \vee & \mathrm{E} & & \mathrm{A}\\ & \wedge & & \vee & \wedge\\ & & \mathrm{B} & & \mathrm{B}\\ \overline{\mathrm{A}} & \vee & \wedge & & \wedge\\ & & \mathrm{C} & & \mathrm{C}\end{array}$$

The reader is referred to Section 3.3 for the formal definitions of CC and of CPE and to [9] for the proofs of the lemmas below.

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 operations that are the duals 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 $H = \{\mathrm{A}, \overline{\mathrm{A}}\}$ be a link, and let $M = (X,\ Y)_{\mathrm{c}}$ 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 $\overline{\mathrm{A}}$ must be (up to commutations and reassociations) arguments of the same conjunction. 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($\overline{\mathrm{A}}$, M) = *false*. Otherwise (i.e., if H consists of two c-blocks),

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

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

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

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($\overline{\mathrm{A}}$, Y), i.e., except those through the link.

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

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 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 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 the Theorem 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 $\psi(\mathrm{FD}(G))$, where

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

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

Much of the material in this section is a detailed description of the results sketched in [14]. Our goal 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 X and Y be the arguments of a disjunction in a semantic graph G, where A_X and A_Y are occurrences of the literal A in X and in Y respectively. Then we call $\{A_X, A_Y\}$ a *disjunctive anti-link*. Note that $M = (X, Y)_d$ 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*.

The following theorem relates subsumed paths to anti-links. The theorem is immediate for CNF formulas; there is an obvious dual theorem regarding subsumed c-paths that is immediate for DNF formulas.

Theorem 4. Let G be a semantic graph in which d-path p is subsumed by a distinct non-tautological d-path p′ in G. Then G contains either a disjunctive anti-link or a conjunctive anti-link.

Proof: There are two distinct possibilities, either $\ell(p) = \ell(p')$ or $\ell(p) \supset \ell(p')$. Suppose $\ell(p) = \ell(p')$. Then there must be a literal having two different occurrences . These two occurrences must be either d- or c-connected and thereby constitute either a disjunctive or a conjunctive anti-link.

Suppose $\ell(p) \supset \ell(p')$. The proof is by induction on the structure of G.

Basis: G is a literal. The result is vacuously true since there cannot be two distinct d-paths through G.

Induction step: Suppose $G = (X \wedge Y)$ for some X and Y. Then either p and p′ are both from the same explicit subgraph (X or Y) or from different explicit subgraphs. If they lie in the same explicit subgraph then the result follows directly from the induction hypothesis. If they are from different subgraphs then every literal in $\ell(p')$ occurs at least once in X and at least once in Y. Any two such occurrences of some literal in $\ell(p')$ constitute a conjunctive anti-link.

Suppose $G = (X \vee Y)$. Let p_X and p_Y be the restriction of p to X and to Y respectively. Let p_X' and p_Y' be the restriction of p′ to X and to Y respectively. Since p and p′ are distinct, either p_X and p_Y must be distinct, or p_X' and p_Y' must be distinct (or both), so assume without loss of generality that p_X and p_X' are distinct.

If either p_X subsumes p_X' or vice versa, then by the induction hypothesis, X must have an anti-link and so does G. On the other hand if p_X and p_X' do not subsume each other, then there must be some literal (say L) in $\ell(p_X')$ which is not in $\ell(p_X)$. But since p′ subsumes p, there must be an occurrence of L in p_Y. The two occurrences of L, one in p_X' and the other in p_Y, constitute a disjunctive anti-link. □

Unfortunately, the presence of anti-links does not imply the presence of subsumed paths, and hence the converse of the above theorem is not true.

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

$$\mathrm{DADV}(\{\mathrm{A}_X, \mathrm{A}_Y\}, M) \quad = \quad \begin{array}{ccccc} \mathrm{B} & \vee & \mathrm{E} & \vee & \mathrm{C} \\ & \wedge & & & \\ \mathrm{B} & \vee & \mathrm{A} & & \\ & \wedge & & & \\ \mathrm{A} & \vee & \mathrm{C} & & \end{array}$$

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

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

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

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

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.

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

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

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

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:

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

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

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

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.

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

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

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

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

Now consider the right hand side of the equation. Since A is in X, DPE(A, G) = DPE(A, X) $\vee$ Y. Therefore, DPE(A, G) − {A} = DPE(A, X) − {A} $\vee$ Y. Also, CE(A) will be disjoint from Y, and thus DC(CE(A), G) = DC(CE(A), X) $\vee$ Y. Therefore we can write the right hand side of the equation as (DPE(A, X) − {A} $\vee$ Y) $\wedge$ (DC(CE(A), X) $\vee$ Y). By factoring out the subgraph Y we get an equivalent subgraph ((DPE(A, X) − {A}) $\wedge$ DC(CE(A), X)) $\vee$ 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 = ($X \wedge Y$); 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) $\wedge$ Y, and CC(A, X) cannot be the empty graph. Therefore, by the induction hypothesis, CC(A, G) = (DPE(A, X) − {A}) $\wedge$ DC(CE(A), X) $\wedge$ 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) $\wedge$ Y. Therefore the right hand side of the equation is (DPE(A, X) − {A}) $\wedge$ (DC(CE(A), X) $\wedge$ 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 $\mathcal{DP}_{A_X,A_Y,G}$.

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 $\mathcal{DP}_{A_X,A_Y,M}$ 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 $\mathrm{DC}(A_X, X) \wedge \mathit{DPE}(A_X, X)$, and from the distributive law,

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

Similarly, Y is equivalent to $\mathrm{DC}(A_Y, Y) \wedge \mathit{DPE}(A_Y, Y)$, and we expand the upper occurrence of Y and distribute.

$$M \;=\; \begin{array}{c} \mathrm{DC}(A_X, X) \;\vee\; \mathrm{DC}(A_Y, Y) \\ \wedge \\ \mathrm{DC}(A_X, X) \;\vee\; \mathrm{DPE}(A_Y, Y) \\ \wedge \\ \mathrm{DPE}(A_X, X) \;\vee\; Y \end{array}$$

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 $\mathcal{DP}_{A_X,A_Y,M}$ have been omitted.

Consider the d-paths of $\mathrm{DC}(A_X, X)$ – the d-paths in X that miss A_X. They either miss $\mathrm{CE}(A_X)$, the c-extension of A_X, or pass through $\mathrm{CE}(A_X) - \{A_X\}$. Hence we have $\mathrm{DC}(A_X, X) = \mathrm{DPE}(\,(\mathrm{CE}(A_X) - \{A_X\}\,), X) \wedge \mathrm{DC}(\mathrm{CE}(A_X), X)$, and d-paths are preserved. By replacing the lower occurrence of $\mathrm{DC}(A_X, X)$ 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

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

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''''.

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

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

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

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, and this is linear in 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. □

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($\{\mathrm{A}_X, \mathrm{A}_Y\}$, 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 $\mathrm{CE}(\mathrm{A}_X) = \mathrm{A}_X$ (and $\mathrm{CE}(\mathrm{A}_X) \neq X$), then $\mathrm{DC}(\mathrm{CE}(\mathrm{A}_X), X) = \mathrm{DC}(\mathrm{A}_X, X)$. Therefore by (possibly non atomic) factoring on $\mathrm{DC}(\mathrm{A}_X, X)$ and observing that $(\mathrm{DC}(\mathrm{A}_Y, Y) \wedge \mathrm{DPE}(\mathrm{A}_Y, Y)) = Y$, DADV($\{\mathrm{A}_X, \mathrm{A}_Y\}$, M) becomes

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

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

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

Since $\mathrm{CE}(\mathrm{A}_X) = \mathrm{A}_X$, the simplified rule for this case results in the following graph.

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

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

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

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

$$A_X \ \vee \ CC(A_Y, Y)$$

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 $CE(A_X) = X$ and $CE(A_Y) = Y$ hold. Under these conditions, $DADV(\{A_X, A_Y\}, M)$ becomes

$$\begin{array}{ccc} X - \{A_X\} & \vee & Y - \{A_Y\} \\ & \wedge & \\ & A & \end{array}$$

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 $X = A_X$, $Y = A_Y$, and $DADV(\{A_X, A_Y\}, M) = A_X$. Note, however, that technically 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 $\{A_X, A_Y\}$, where the smallest full block *M* containing the anti-link is $(A_X \wedge Y)$. 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 $DC(A_Y, Y)$.

This is a kind of dual to Case 3 of the simplified versions of DADV discussed earlier. There, the anti-link $\{A_X, A_Y\}$ is disjunctive and $M = (A_X \vee Y)$. The simplified DADV operation just replaces *Y* by $CC(A_Y, Y)$. 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. Complexity Considerations

The problem of eliminating all subsumed paths in a graph in an efficient manner does not seem feasible. Let *G* be a semantic graph; we say that a graph G' is a *d-minimal equivalent* of *G* if it satisfies the following conditions.

1. *G* is logically equivalent to G'.
2. If p′ and q′ are two distinct d-paths in G', then p′ does not subsume q′ and vice versa.
3. If p′ is a d-path in G', then there is a d-path *p* in *G* such that, $\ell(p) = \ell(p')$.
4. If p is a minimal d-path in *G*, then there is a d-path p′ in G' such that $\ell(p') = \ell(p)$.

Note that Property 1 is implied by Properties 3 and 4, and that G' need not be unique. However, the d-paths of G' will always include all essential (and possibly some inessential) prime implicates of *G*. The *c-minimal equivalent* of a graph is defined in the obvious dual way.

Computing d-minimal equivalent graphs efficiently would be helpful for finding prime implicates. In a d-minimal equivalent graph of a full dissolvent, subsumption checks can be completely eliminated by Property 2 above. Hence to find the prime implicates of *G*, we can find a d-minimal equivalent G' of the full dissolvent FD(*G*), and then simply enumerate the d-paths of G'.

A d-minimal equivalent of a given graph G can be trivially obtained by first enumerating all the d-paths of the given graph G and then eliminating all the subsumed d-paths. The above algorithm is exponential in the size of G, because G' is being constructed in CNF. However an NNF d-minimal equivalent G' of G may be small compared to a CNF d-minimal equivalent. Even so, the problem is NP-hard (proof follows) and hence is not likely to have an efficient algorithm.

Theorem 8. The following problem (*Elimination of subsumed paths*) is NP-hard. Given a graph G, find a d-minimal equivalent graph G'.

Proof: To show NP-hardness we reduce from satisfiability of CNF formulas. Let C be an instance of the CNF satisfiability problem and $\{X_1, X_2, \cdots, X_n\}$ be the set of variables in C. Let $A, X_1', X_2', \cdots, X_n'$ be variables not occurring in $\{X_1, X_2, \cdots, X_n\}$. Let $\mathcal{D}$ be the semantic graph obtained by replacing $\overline{X_i}$ by X_i', $1 \le i \le n$, in $\neg C$. We construct the following semantic graph G.

$$\begin{array}{ccc} A & \vee & \mathcal{D} \\ & \wedge & \\ X_1 & \vee & X_1' \\ & \wedge & \\ & \vdots & \\ & \wedge & \\ X_n & \vee & X_n' \end{array}$$

The size of the graph G is no more than a constant factor of the size of C and can therefore be constructed in linear time. It is easy to see that any d-path which includes the literal A must pass through $\mathcal{D}$ and vice versa.

Let G' be any graph which is a d-minimal equivalent to G. We will show that C is satisfiable iff the literal A occurs in G'.

Suppose C is satisfiable; then $\neg C$ is falsifiable, and there are d-paths (in fact, clauses, since $\neg C$ is in DNF) in $\neg C$ that do not contain any disjunctive link $\{X_i, \overline{X_i}\}$. All such d-paths through $\mathcal{D}$ do not contain any $\{X_i, X'_i\}$, $1 \le i \le n$; at least one such, say p, is not subsumed by another d-path through $\mathcal{D}$. The d-path pA cannot be subsumed by any other d-path in G and hence there will be a path in G' which has the same literal set as pA. Hence the literal A must occur in G'.

If C is not satisfiable, then $\neg C$ is valid. Therefore for every d-path p in $\mathcal{D}$ (and hence every d-path through A), there is some i, $1 \le i \le n$, such that the pair of literals

$\{X_i, X'_i\} \subseteq \ell(p)$. But every such pair of literals forms a d-path in G and hence every d-path containing A will be subsumed by another d-path in G. Furthermore the subsuming path will not contain the literal A. By definition of G', no d-path in G' can contain the literal A, and thus the literal A cannot occur in G'.

If we can solve the elimination of subsumed paths problem in polynomial time then we have the following algorithm which can solve the satisfiability of CNF in polynomial time. Given any instance C of the CNF satisfiability problem, we can construct in polynomial time the graph G as shown earlier. We then find the graph G' using the algorithm for elimination of subsumed paths. The size of G' will be polynomial in the size of G (since computing it required only polynomial time). Now C is satisfiable iff the literal A occurs in G' and this check can be done in polynomial time. □

Corollary. Given a graph G, finding a c-minimal equivalent of G is NP-hard.

We have seen that the general problem of computing d- or c-minimal graphs is NP-hard. Nevertheless, redundant disjunctive anti-links are easily recognized, and eliminating their corresponding subsumed d-paths can be done without direct subsumption checks. On the other hand, recognizable subsumed d-paths due to conjunctive anti-links are not likely to be as plentiful due to the strong restriction defining such useful anti-links. It is also difficult to find out if an arbitrary conjunctive anti-link results in subsumed d-paths. In fact, this problem is NP-complete.

Theorem 9. The following problem is NP-complete.

INSTANCE: Given a conjunctive anti-link $\{A_X, A_Y\}$ in graph G.

QUESTION: Are there two d-paths p_X and p_Y in G, such that p_X passes through A_X and p_Y passes through A_Y and either p_X subsumes p_Y or vice versa.

Proof: It is easy see that this problem is in NP. To show NP hardness we reduce from satisfiability of CNF formulas. Let C be an instance of the CNF satisfiability problem and $\{X_1, X_2, \cdots, X_n\}$ be the set of variables in C. Let $A, B, X_1', X_2', \cdots, X_n'$ be variables not occurring in $\{X_1, X_2, \cdots, X_n\}$. Let $\mathcal{D}$ be semantic graph obtained by replacing $\overline{X_i}$ by X_i', $1 \le i \le n$, in $\neg C$. We construct the following semantic graph G.

$$
\begin{array}{ccccccccccc}
A_1 & \vee & \mathcal{D} & & & & & & & & \\
 & \wedge & & & & & & & & & \\
 & & X_1 & & X_2 & & & & & & X_n \\
A_2 \vee B & \vee & \wedge & \vee & \wedge & \vee & \cdots & \vee & & & \wedge \\
 & & X_1' & & X_2' & & & & & & X_n'
\end{array}
$$

The subgraph $\mathcal{D}$ is a DNF formula, and A_1 and A_2 are two different occurrences of the literal A. These two literal occurrences form a conjunctive anti-link in G. Every d-path through A_2 contains the literal B, and only d-paths containing A_1 can possibly subsume d-paths through A_2. The size of the graph G is no more than a constant factor of the size of C and can therefore be constructed in linear time.

We must show that C is satisfiable iff there is a d-path through A_1 that subsumes another d-path through A_2. Suppose first that C is satisfiable. Then $\neg C$ is falsifiable, and there is at least one d-path in $\neg C$ that does not contain any disjunctive link $\{X_i, \overline{X_i}\}$. Therefore some d-path p through $\mathcal{D}$ does not contain any of the literal pairs $\{X_i, X_i'\}$, $1 \le i \le n$. Recall that $\ell(p)$ is defined to be the literal set of path p. It is easy to see that since $\ell(p)$ does not contain such a literal pair (corresponding to a d-link in $\neg C$), a d-path p′ can be chosen that passes through the n rightmost disjuncts in the lower part of G, such that $\ell(p') \supseteq \ell(p)$. Clearly, A_2Bp' is subsumed by A_1p.

To show the if part, suppose there is some d-path (say p) through A_1 that subsumes another d-path (say p′) through A_2. Then p cannot contain any literal pair $\{X_i, X_i'\}$ because there no such pair occurs on any d-path in the lower part of G. Therefore p when restricted to $\mathcal{D}$ will not have such a literal pair, $\neg C$ has a d-path without a d-link, and hence C is satisfiable. □

Corollary. The problem dual to Theorem 9 involving disjunctive anti-links and c-paths is NP-complete.

5. Benchmark Examples

Ngair [10] has investigated examples that prove difficult for many proposed prime implicate/implicant algorithms. In this section, we show that PI + anti-links is effective for some of these examples. For other examples from [10], applying anti-link techniques appears not to produce as significant an improvement. We develop an additional technique based on *strictly pure* full blocks that results in a dramatic improvement for these latter examples.

5.1. Examples Using Anti-Links

In [10] Ngair proposes a class of formulas for which reliance on an intermediate CNF form can result in an exponential increase in size and hence would be intractable for CNF-based algorithms. Dissolution + PI also does poorly for these examples: Although the full dissolvent can be computed quickly, a large number of subsumption checks must be performed by PI. It turns out, however, that the subsumed implicates correspond to useful and easily recognizable anti-links of both the disjunctive and conjunctive kind. We show that if these anti-links links are removed after dissolution is performed, dissolution + PI can find all the implicates in polynomial time.

We represent this class of formulas as $\{\mathcal{F}_i\}$, $i \geq 1$. Below we show the graph for $\mathcal{F}_N$:

$$
\begin{array}{ccccc}
 & \begin{array}{c} \overline{A_3} \\ \wedge \\ \vdots \\ \wedge \\ \overline{A_{2N-1}} \end{array} & \vee & A_1 & \\
 & & \wedge & & \\
 & \begin{array}{c} \overline{A_4} \\ \wedge \\ \vdots \\ \wedge \\ \overline{A_{2N}} \end{array} & \vee & A_2 & \\
 & & \wedge & & \\
\begin{array}{c} A_1 \\ \wedge \\ A_2 \end{array} & \vee & \cdots & \vee & \begin{array}{c} A_{2n-1} \\ \wedge \\ A_{2n} \end{array}
\end{array}
$$

Clearly $\mathcal{F}_N$ has $3 \cdot N$ literals, and $2 \cdot N$ c-links; dissolution can remove these links by performing $2 \cdot N$ dissolution steps. The full dissolvent that results is:

$$
\begin{array}{ccccccccc}
\overline{A_3} & & & & & & & & \\
\wedge & & & & & & & & \\
\vdots & \vee & A_1 & & A_1 & & & & A_1 \\
\wedge & & & & \wedge & & & & \wedge \\
\overline{A_{2N-3}} & & & & A_2 & & & & A_2 \\
 & \wedge & & \vee & \wedge & \vee & \cdots & \vee & \wedge \\
A_1 & & A_1 & & A_{2N-1} & & & & A_3 \\
\wedge & \vee & \wedge & & \wedge & & & & \wedge \\
A_2 & & A_2 & & A_{2N} & & & & A_2 \\
 & & \wedge & & & & & & \\
 & & \overline{A_4} & & & & & & \\
 & & \wedge & & & & & & \\
 & & \vdots & & & & & & \\
 & & \wedge & & & & & & \\
 & & \overline{A_{2N}} & & & & & &
\end{array}
$$

Although the structure of the full dissolvent depends on the order in which dissolution was applied, the above dissolvent is the one obtained by the current version of our propositional dissolution prover Dissolver. (The first of the two compact versions of the dissolvent from Section 2.2 is used; *X* is chosen to be the smallest of the two c-blocks). We can now factor on all the occurrences of both A_1 and A_2 in the upper right hand part of the graph and on the two occurrences of A_1 in the lower left corner. The resulting graph is shown below. (Since Dissolver is NNF-based, such factoring is not only feasible but is in fact implemented and routinely employed to produce the final output.)

$$\begin{array}{ccc}
\overline{A_3} & & \\
\wedge & & \\
\vdots & \vee & A_1 \\
\wedge & & \\
\overline{A_{2N-3}} & & \\
 & \wedge & \\
 & A_1 & \\
 & & A_2 \\
 & \wedge & \wedge \\
 & & \overline{A_4} \\
A_2 & \vee & \wedge \\
 & & \vdots \\
 & & \wedge \\
 & & \overline{A_{2N}}
\end{array}
\quad \vee \quad
\begin{array}{ccc}
 & A_1 & \\
 & \wedge & \\
 & A_2 & \\
 & \wedge & \\
A_{2N-1} & & A_3 \\
\wedge & \vee \cdots \vee & \wedge \\
A_{2N} & & A_2
\end{array}$$

The two occurrences of A_2 at the bottom left part of the graph form a redundant disjunctive anti-link; they can be removed using the special case covered by Rule 1 for disjunctive anti-links. The two occurrences of A_1 on the left hand side of the graph form a conjunctive anti-link and can be removed using the conjunctive anti-link rule. This produces:

$$\begin{array}{c}
A_1 \\
\wedge \\
A_2
\end{array}
\quad \vee \quad
\begin{array}{ccc}
 & A_1 & \\
 & \wedge & \\
 & A_2 & \\
 & \wedge & \\
A_{2N-1} & & A_3 \\
\wedge & \vee \cdots \vee & \wedge \\
A_{2N} & & A_2
\end{array}$$

By factoring on A_1 and removing the conjunctive anti-link comprised of the two occurrences of A_2 (or by just factoring on $A_1 \wedge A_2$), the above graph reduces to $(A_1 \wedge A_2)$; the prime implicates are just $\{A_1\}$ and $\{A_2\}$. To get this graph, N+3 factoring and 3 anti-link operations were required – obviously polynomial time. Hence dissolution + removal of anti-links + PI can handle the above class of problems in polynomial time. Perhaps the most important point is that no subsumption checks whatsoever are required.

5.2. Strictly Pure Full Blocks

Recall that a full block is essentially an explicit subgraph; it is a subset of the arguments of a conjunction or disjunction, and, via commutations and reassociations, can in fact be made explicit. We say a full block M in graph G is *pure* if there are no c-links or d-links that consist of exactly one node from M. (This is just the obvious generalization of pure literal used in the literature on CNF-based automated deduction.) If, in addition, there are no conjunctive or disjunctive anti-links that consist of exactly one node from M, we say that M is *strictly pure*. (Simply put, M shares no variables with the rest of G.) Observe that pure and strictly pure are also well defined for arbitrary subgraphs, not just for full blocks.

When factored, some of the examples from [10] contain surprisingly many strictly pure full blocks. Note that both factoring and recognizing strictly pure full blocks are polynomial operations. Intuitively, such full blocks can be replaced by single new variables, and the implicates of the resulting graph bear a strong relationship to those of the original. Of course, the full block in question must be *satisfiable* (since the new variable certainly is). At first, this may appear to be a heavy penalty. It is not, however, because the prime implicates of the full block itself must be computed anyway. In doing so, its satisfiability is determined as a byproduct.

Theorem 10. Let M be a satisfiable strictly pure full block in satisfiable semantic graph G, and let X be a new variable not occurring in G. We denote by G_M the subgraph of G relative to nodes in M, and by G_{X} the graph obtained by the substitution of X for M in G. Let $D = D_{G-M} \vee D_M$ be a non-tautological disjunction of literals from G, where D_{G-M} consists of literals not in M, and D_M consists of literals that are. We define the disjunction D_{X} to be $D_{G-M} \vee \{D_M / \mathrm{X}\} = D_{G-M} \vee \mathrm{X}$ (the result of replacing in D every literal of D_M by the new variable X and collapsing the multiple occurrences of X). Then

$$G \models D \text{ and } G \not\models D_{G-M} \quad \Leftrightarrow$$
$$G_{\mathrm{X}} \models D_{\mathrm{X}} \text{ and } G_M \models D_M \text{ and } G_{\mathrm{X}} \not\models D_{\mathrm{X}} - \{\mathrm{X}\}.$$

Proof: For the ''only if'' part, assume that $G \models D$ and that $G \not\models D_{G-M}$. We first show that $G_{\mathrm{X}} \models D_{\mathrm{X}}$. Observe that $G \wedge \overline{D} = G \wedge \overline{D_{G-M}} \wedge \overline{D_M}$ is unsatisfiable and hence spanned by its conjunctive links. Therefore, every satisfiable c-path in G must be linked to $\overline{D}$. (Since D is non-tautological, every c-path p through G must either contain a link or be linked to a literal in $\overline{D}$.)

Consider an arbitrary satisfiable c-path p in G. As with D, p consists of both literals from M and literals not from M: We write $p = p_{G-M}\, p_M$, where either p_{G-M} or p_M may be empty, but not both. Corresponding to p, there is a c-path p_X in G_X, where $p_X = p_{G-M}\, X$ if p_M is non-empty, otherwise $p_X = p_{G-M}$. Note that this defines a one-to-many onto mapping from c-paths of G_X to those of G and that $\ell(p_X) = \ell(p_{G-M}) \cup \{X\}$ if p_M is non-empty, otherwise $\ell(p_X) = \ell(p_{G-M})$.

Since M is a full block, p must either go through M or miss M entirely. If p misses M, then p must be linked to $\overline{D_{G-M}}$. Obviously, p_X contains the literals of p_{G-M} and is linked to $\overline{D_X}$. If p goes through M, then p_X contains X and $\overline{D_X}$ contains $\overline{X}$, providing the required link. Therefore, we have shown that $G_X \models D_X$.

To prove that $G_M \models D_M$, first observe that there must exist a satisfiable c-path $p = p_{G-M}\, p_M$ in G that is not linked to $\overline{D_{G-M}}$ (otherwise, we would have $G \models D_{G-M}$, contrary to hypothesis). Therefore, p is linked only to $\overline{D_M}$, and p_M is non-empty. Now consider c-path p′ obtained from p by replacing p_M by some arbitrary satisfiable c-path p_M' in M. Clearly, p′ is also satisfiable and linked only to $\overline{D_M}$. It follows that every satisfiable c-path p_M' in M is linked to $\overline{D_M}$, and hence $G_M \models D_M$.

To prove that $G_X \not\models D_X - \{X\}$, let p be defined, as above, to be a satisfiable c-path in G not linked to $\overline{D_{G-M}}$. Since, by construction, p_X is linked only to $\overline{X}$ in $\overline{D_X}$, G_X cannot entail $D_X - \{X\}$.

The proof of the ‘‘if’’ part of the theorem is similar and is left to the reader. □

Corollary. If $X \in D_X$, then

$$D \text{ is a prime implicate of } G \iff D_X \text{ is a prime implicate of } G_X, \text{ and } D_M \text{ is a prime implicate of } M.$$

If graph G contains several strictly pure full blocks, then the repeated application of Theorem 10 and its corollary provides a potentially significant speedup in computing the prime implicates of G: Replace each strictly pure full block M_i by a new variable X_i and compute the prime implicates of the resulting graph G_X. Then, all substitutions of prime implicates of M_i for X_i in the prime implicates of G_X result in prime implicates of G. The speedup is potentially dramatic: Each subsumption test performed within some M_i would otherwise be performed once for *every* d-path in $G - M_i$ that can be extended through M_i to form a d-path in G. Observe that prime implicates of G_X containing none of the X_i's are simply prime implicates of G that do not contain literals from any of the M_i's.

Several *distinct* strictly pure full blocks can be handled by the repeated application of Theorem 10 as explained above. However, multiple occurrences $M^1, \cdots, M^n$ of a single full block M in G require a separate analysis. The problem is that the multiple occurrences themselves preclude any single M^i from being strictly pure, even if M shares no variables with the rest of G. Intuitively, we would expect that by replacing each of the M^i's in G by the single new variable X, the prime implicates of the resulting graph G_X would also bear a strong relationship to those of G. This is made precise in the next section.

5.3. Multi-Pure Full Blocks

Suppose that $M^1, \cdots, M^n$ are full blocks in G and that all of the M^i's are syntactically equivalent. Suppose further that the subgraph M^* formed by taking all the nodes of the M^i's is not a full block (hardly surprising) but that M^* is strictly pure. We call the M^i's *multi-pure full blocks*. Note that each M^i is *not* strictly pure, since it has anti-links (and possibly links) to its equivalent full blocks in M^*. Since the M^i's are alike syntactically, we use M to denote any full block having their form.

Suppose further that there are occurrences $\overline{M^{n+1}}$, $\overline{M^{n+2}}$, ..., $\overline{M^{n+m}}$ of full block M. Thus, we allow for the presence of *complementary multi-pure full blocks*. (We may consider $\overline{M_i}$ to be the NNF form of $\neg M$; alternatively, when given a formula not in NNF and in which there are occurrences of both M and $\neg M$, a system may simply note the the non-atomic complementarity. In either case, we do not address the issue of recognizing such complementary non-atomic formulas.) It turns out that the results of Theorem 10 can be extended to the case in which a formula contains such complementary multi-pure full blocks:

Theorem 11. Let M^1, M^2, ..., M^n and $\overline{M^{n+1}}$, $\overline{M^{n+2}}$, ..., $\overline{M^{n+m}}$ be occurrences of full block M in satisfiable semantic graph G, where all of the M_i's and $\overline{M_j}$'s are satisfiable. Since the M^i's are alike syntactically, we use M to denote any full block having their form; in particular, we treat $\overline{M^i}$ as a negated occurrence of M. Let X be a new variable not occurring in G, and let M^* be the subgraph of G relative to the M^i's. We denote by G_X the graph obtained by the substitution of X for M in G. Let $D = D_{G-M} \vee D_M$ be a non-tautological disjunction of literals from G, where D_{G-M} consists of literals not in M^*, and D_M consists of literals that do come from M^*. We define the disjunction D_X to be $D_{G-M} \vee \{D_M / X\} = D_{G-M} \vee X$, the result of replacing in D every literal of D_M by the new variable X and collapsing the multiple occurrences of X. Similarly we define $D_{\overline{X}} = D_{G-M} \vee \{D_M / \overline{X}\} = D_{G-M} \vee \overline{X}$. (Intuitively, we use D_X when the literals of D_M correspond to unnegated occurrences of M, and $D_{\overline{X}}$ for negated occurrences of M.) Then

D is a prime implicate of G $\Leftrightarrow$

$$\left[\begin{array}{c} D_M = \varnothing \text{ and } D \text{ is a prime implicate of } G_X \\ \text{OR} \\ D_M \neq \varnothing \text{ and } D_X \text{ is a prime implicate of } G_X \text{, and } D_M \text{ is a prime implicate of } M \\ \text{OR} \\ D_M \neq \varnothing \text{ and } D_{\overline{X}} \text{ is a prime implicate of } G_X \text{ ,} D_M \text{ is a prime implicate of } \overline{M} \end{array}\right]$$

Proof:. Let G'_X be a graph without c-links that is equivalent to G_X. (E.g., G'_X could be the full dissolvent of G_X). Similarly, let M' be a linkless equivalent of M, and $\overline{M}'$ be a linkless equivalent of $\overline{M}$. Let G' be the graph obtained from G by replacing X by M' and $\overline{X}$ by $\overline{M}'$. It is easy to see that G' is equivalent to G but has no c-links. By Theorem 3, every prime implicate of G is present as a d-path in G' and every prime implicate of G_X is present as a d-path in G'_X.

To prove the only if part, let D be a prime implicate of G. Then there must an unsubsumed d-path p in G' such that $\ell(p) = D$. Since M' and $\overline{M}'$ are complementary, p can never meet (and thus pass through) both M_i and $\overline{M}_j$ for any i and j.

Suppose first that p does not pass through any of the M's or $\overline{M}$'s. Obviously,p must also be a d-path in G'_X. (Technically p is isomorphic to a d-path in G'_X.) To prove that p is not subsumed by another d-path in G'_X, assume otherwise, i.e., that p' in G'_X subsumes p. But p' cannot contain X or $\overline{X}$, and hence would also be a d-path in G' that subsumes p, which is a contradiction. Thus D is a prime implicate of G'_X.

Suppose p passes through the full blocks $M^{i_1}{}' \cdots M^{i_q}{}'$. Let p_j, $1 \le j \le q$, be the restriction of p to $M^{i_j}{}'$. Notice that for $1 \le j,k \le q$, $\ell(p_j) = \ell(p_k)$; otherwise, the d-path obtained by replacing p_j by p_k in p would subsume p. Similarly, p_j cannot be subsumed by another d-path in $M^{i_j}{}'$. Since p_j is an unsubsumed d-path in $M^{i_j}{}'$, a linkless equivalent of M, $D_M = \ell(p_j)$ is a prime implicate of M. Now let p_X be the d-path obtained by replacing each of the p_j's by X in p; p_X is a d-path in G_X. Furthermore, p_X cannot be subsumed by another d-path in G_X (again, such a subsuming path would induce a path in G' that subsumes p). Therefore $D_X = \ell(p_X)$ is a prime implicate of G_X.

In the case that p passes through the full blocks $\overline{M}^{i_1}{}'$ ... $\overline{M}^{i_r}{}'$, the argument is essentially the same.

To prove the if part, first suppose $D_M = \emptyset$ and D is a prime implicate of G_X. Then there is an unsubsumed d-path (say p) in $G_X{}'$ such that $D = \ell(p)$. Since $D_M = \emptyset$, p contains neither X nor $\overline{\mathrm{X}}$. Thus p is an unsubsumed d-path of G', and D is a prime implicate of G.

Now suppose that D_X is a prime implicate of G_X and that D_M is a prime implicate of M. Then there is an unsubsumed d-path p_X in $G_X{}'$ and there is an unsubsumed d-path p_M in M' such that $\ell(p_X) = D_X$ and $\ell(p_M) = D_M$. By definition, D_X contains X (and cannot contain $\overline{\mathrm{X}}$). Let p' be the result of replacing all occurrences of X in p_X by p_M; p' is a d-path in G'. Since both p_M and p_X are unsubsumed in G_X and M' respectively, and since M' does not share any variables with the rest of G', p' will also be unsubsumed in G'. Hence by Theorem 3, $D = \ell(p')$ is a prime implicate of G.

Similarly, if $D_{\overline{X}}$ is a prime implicate of G_X and D_M is a prime implicate of $\overline{M}$, then D is a prime implicate of G. □

We expect, of course, to substitute new variables for such multi-pure full blocks and achieve a savings in the computation of prime implicates comparable to that provided by Theorem 10. But note that some complementary occurrences of M may be c-connected; this means that such occurrences play a role in whatever inference process is employed prior to computation of the implicates themselves. In particular, we may treat them as literals and dissolve (indeed, the set of individual links between such full blocks would satisfy the requirements of a multiple link dissolution chain).

That dissolving on two complementary full blocks accomplishes exactly what the corresponding single-link dissolutions would is clear: All c-paths through both full blocks are eliminated from the graph. But the former operation is much more efficient than the latter. Therefore, recognizing such complementary full blocks and performing inference directly on them, rather than on their constituent literals, is desireable. Note also that for the inference phase of a prime implicate computation, complementary full blocks do *not* have to be multi-pure full blocks. This is necessary only for the extraction of implicates using Theorem 11 once all implicates are known to be present.

Finally, the remarks above apply also to identical full blocks if they form appropriate non-atomic anti-links as discussed in Section 3.

5.4. More examples

In [10] Kean provides the class of examples referred to as K_{mn}. They have $m(n+1)$ input CNF clauses and $(m+1)^n + m\,n$ prime implicates. This set of clauses can be factored to get a more compact representation in NNF as shown below.

$$
\begin{array}{ccc}
 & & \overline{S_{1\,1}} \\
 & & \wedge \\
A_1 & \vee & \vdots \\
 & & \wedge \\
 & & \overline{S_{1\,m}} \\
 & \wedge & \\
 & \vdots & \\
 & \wedge & \\
 & & \overline{S_{n\,1}} \\
 & & \wedge \\
A_n & \vee & \vdots \\
 & & \wedge \\
 & & \overline{S_{n\,m}} \\
 & \wedge & \\
\overline{A_1} & \vee \cdots \vee & \overline{A_n}
\end{array}
$$

Since the number of prime implicates is exponential, so is the number of subsumption checks required. The number of subsumption checks for the IPIA [1] and GEN-PI [10] are shown in Table 1.

Examples	IPIA[1]	GEN–PI[2]	PI+ANTI-LINK
$K_{3\,3}$	5166	972	164
$K_{4\,4}$	506472	11600	887
$K_{5\,4}$	1730120	29074	887

Table 1.

For each i, the literals $\{\overline{S_{i\,1}}\ \cdots\ \overline{S_{i\,m}}\}$ form a full block M_i, and all literals in it are strictly pure. Let K'_{mn} be the graph obtained by replacing each full block M_i by a new variable X_i. By the corollary of Theorem 10, we can get the prime implicates of K_{mn} from the prime implicates of K'_{mn}. Since each of the subgraphs M_i has no c-links, the prime implicates of M_i are present as d-paths by Theorem 3. Since they also have no anti-links, by the contrapositive of Theorem 4, neither are subsumption checks required to find these prime implicates. Thus the number of subsumption checks to be done is exactly that required for computing the prime implicates of K'_{mn}, and this is significantly less than that needed for K_{mn}. Note that the number of prime implicates of K'_{mn} is only 2^n+n. For the problems in Table 1, we applied the above technique in combination with anti-link operations. For K'_{mn}, the full dissolvent depends only on n and can be defined recursively. The full dissolvent of K'_{m2} (basis) is shown below on the left; the general case of K'_{mn} is shown on the right.

$$
\begin{array}{c} A_1 \\ \wedge \\ \overline{A_2} \\ \wedge \\ \overline{X_2} \end{array}
\ \vee\
\begin{array}{c} \overline{A_1} \\ \wedge \\ X_1 \\ \wedge \\ A_2 \vee X_2 \end{array}
\qquad\qquad
\begin{array}{c} X_n \vee A_n \\ \\ \wedge \\ \\ K'_{m\,n-1} \end{array}
\ \vee\
\begin{array}{c} X_n \\ \wedge \\ \overline{A_n} \\ \wedge \\ A_1 \\ A_2 \vee X_2 \\ \vdots \\ A_n \vee X_n \end{array}
$$

The number of subsumption checks required here is also shown in Table 1. There is clearly a significant reduction in the number of subsumption checks required by our techniques. Note that for the problem $K_{m\,n}$ the number of subsumption checks depends only on n and not on m.

Our techniques are not limited to NNF formulas. They can sometimes be used by other algorithms like IPIA and GEN-PI which are not based on NNF formulas. For example K'_{mn} turns out to be in CNF and hence both IPIA and GEN-PI can handle these formulas, thereby reducing the number of subsumption checks needed. However normal forms like CNF provide very little scope for applying these techniques directly. For example the literals $\{\overline{S_{i\,1}}\ \cdots\ \overline{S_{i\,m}}\}$ in the unfactored form of $K_{m\,n}$ do not form a

---

[1] Data obtained from [1].

[2] Inferred from [10].

full block. Hence one cannot apply Theorem 10. They do form a full block after factoring. This provides stronger evidence that by avoiding less general normal forms like CNF/DNF, one can improve the performance of prime implicate algorithms.

We also note that we have implemented some of the anti-link operations presented above in a diagnosis system introduced in [13]. As a result, we greatly increased the size of problems solvable on our system. (The anti-link operations used were the special cases that do not increase the size of the formula.)

6. Conclusions 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.

Although prime implicate/implicant problems are intractable in general, our techniques perform exponentially better than others on certain examples. In addition, we are able to improve performance greatly on the inherently exponential examples of [10].

Some experimental results on a dissolution- and PI-based prime implicate system are reported in [12]. The system is currently being extended to include operations based on anti-links and strictly pure full blocks, so as to test their effectiveness in practice.

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 the 10th International Conference on Automated Deduction*, Kaiserslautern, W. Germany, July, 1990. In *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Vol. 449, 543-557.
3. Jackson, P. Computing prime implicants incrementally. *Proceedings of the 11th International Conference on Automated Deduction*, Saratoga Springs, NY, June, 1992. In *Lecture Notes in Artificial Intelligence*, 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 the 6th National Conference on Artificial Intelligence*, 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 the 4th *International Conference on Logic Programming and Automated Reasoning*. St. Petersburg, Russia, July 13-20,1993. In *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Vol. 698, 277-288.

13. Ramesh, A., and Murray, N.V. An application of non clausal deduction in diagnosis. Technical Report 95-1, Dept. of Computer Science, SUNY at Albany, January, 1995.

14. Ramesh, A., Beckert, B., Hähnle, R., and Murray, N.V. ‘‘On Anti-Links.’’ *Proceedings* of the *5*th *International Conference on Logic Programming and Automated Reasoning*, Kiev, Ukraine, July 16-21, 1994. In *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Vol. 822, 275-289.

15. Reiter, R. and de Kleer, J. Foundations of assumption-based truth maintenance systems: preliminary report. *Proceedings of the 6th National Conference on Artificial Intelligence*, Seattle, WA, July 12-17, 1987, 183-188.

16. 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.

17. Strzemecki, T. Polynomial-time algorithms for generation of prime implicants. *Journal of Complexity* **8** (1992), 37-63.