Fast Subsumption Checks Using Anti-Links[*]

Anavai Ramesh[†]
Intel Corporation, MS CH6-418, 5000 W. Chandler Blvd., Chandler, AZ 85226.
Email: agramesh@sedona.intel.com

Bernhard Beckert and Reiner Hähnle
University of Karlsruhe, Institute for Logic, Complexity and Deduction Systems, 76128 Karlsruhe, Germany. Email: {beckert,haehnle}@ira.uka.de,
WWW: http://i12www.ira.uka.de/

and

Neil V. Murray
Institute for Programming & Logics, Department of Computer Science, University at Albany, Albany, NY 12222. Email: nvm@cs.albany.edu

January 31, 1996

Abstract. The concept of *anti-link* is defined (an anti-link consists of two occurrences of the same literal in a formula), 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 that generalize the concept of *pure literals* 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.

1 Introduction

The logical consequences of a ground formula, expressed as minimal implied clauses, are useful in certain approaches to non-monotonic reasoning (Kean and Tsiknis, 1992; Przymusinski, 1989; de Kleer *et al.*, 1992), where all consequences of a formula set (e.g., the support set for a proposed common sense conclusion) are required. Minimal conjunctions that imply a formula are useful in situations where satisfying models are desired, as in diagnosis (de Kleer and Williams, 1987; Mozetič and Holzbaur, 1994) and logic design (Sasao, 1993). Such minimal implied clauses are the formula's *prime implicates*, and the minimal conjunctions that imply it are its *prime implicants*.

[*] This research was supported in part by National Science Foundation Grant CCR-9101208 (Ramesh and Murray) and by Deutsche Forschungsgemeinschaft within the Schwerpunktprogramm Deduktion (Beckert and Hähnle). Some of the results in this paper appear in condensed form in the *Proceedings* of the *5th International Conference on Logic Programming and Automated Reasoning (LPAR)*, Kiev, Ukraine, July 16–21, 1994, and in the *Proceedings* of the *National Conference on Artificial Intelligence*, Seattle, WA, July 31–August 4, 1994.

[†] This work has been carried out while the author stayed at the University at Albany.

Many algorithms have been proposed to compute the prime implicates of propositional boolean formulas. Most algorithms (de Kleer, 1992; Jackson and Pais, 1990; Jackson, 1992; Kean and Tsiknis, 1990; Slagle *et al.*, 1970) assume that the input is either in conjunctive normal form (CNF) or in disjunctive normal form (DNF). The algorithm of (Ngair, 1993) requires the input to be a conjunction of DNF formulas. In (Ramesh and Murray, 1993) 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 (Murray and Rosenthal, 1987b), 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 (Ngair, 1993). Coudert and Madre (1993) have also developed an algorithm for computing prime implicates and implicants of formulas in which binary decision diagrams (BDDs) are employed. In that case anti-link operators can be helpful, too: Any algorithm which uses BDDs—including Coudert and Madre's—must perform large amounts of subsumption testing; therefore, it is of advantage to remove anti-links from a formula before it is transformed into a BDD.

Although we use prime implicate/implicant generation as an example to demonstrate the utility of anti-link operations, anti-link operations can be employed in any application that requires eliminating subsumed paths in an NNF formula. We have successfully used anti-link operations in a *diagnosis* system (Ramesh and Murray, 1995).

In (Ramesh and Murray, 1993) 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 (de Kleer, 1992), 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 (*anti-links* consist of two (*non-complementary*) occurrences of the same literal—whereas *links* consist of two *complementary* literals). We identify operations to remove such anti-

[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 *et al.*, 1992). The anti-link rule as it will be defined later can be viewed as an implementation of the regularity condition in (Letz *et al.*, 1992) for the propositional non-clausal case (Letz et al. 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.

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 (1993) is explored. In Section 6 we introduce *strictly pure full blocks* and use them to develop a method that reduces the number of subsumption tests required.

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 negations and other operators 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 (Murray and Rosenthal, 1993). 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

Semantic graphs are a graphical representation of NNF formulas:

DEFINITION 1. A *semantic graph* consists either of

1. one of the constants *true* and *false*,

2. a literal A or $\overline{A}$,

3. a *c-arc*, which is a conjunction of two semantic graphs, or

4. a *d-arc*, which is a disjunction of two semantic graphs.

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.

Each semantic graph used in the construction of a semantic graph G is called an *explicit subgraph* of G. If $G = (X, Y)_c$, then X (resp. Y) is a *fundamental subgraph* of G if X (Y) is not a c-arc; otherwise the fundamental subgraphs of X (Y) are fundamental subgraphs of G. Similarly if $G = (X, Y)_d$, then X (Y) is a *fundamental subgraph* of G if X (Y) is not a d-arc, otherwise the fundamental subgraphs of X (Y) are fundamental subgraphs of G.

The set of *nodes* of a semantic graph G consists of all literal occurrences used in its construction; the same holds for the *set of c-arcs* of G and the *set of d-arcs* of G; i.e., these sets include the nodes, c-arcs, and d-carc, respectively, occurring in the explicit subgraphs of G.

In the following, we identify a semantic graph G and the formula it represents[2]; essentially, the only difference between the semantic graph and the formula is the point of view, and we will use either term depending upon the desired emphasis. For a more detailed exposition, see (Murray and Rosenthal, 1993).

In addition, we identify a semantic graph and the triple consisting of its set of nodes, its set of c-arcs, and its set of d-arcs. The only exception, were this latter identification is not possible, because it would be ambiguous, are the semantic graphs *true* and *false* (both correspond to $(\emptyset, \emptyset, \emptyset)$). Note, however, that when a semantic 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.

In pictorial representations, c-arcs and d-arcs are indicated by the usual symbols for conjunction and disjunction; the arguments of a c-arc are placed vertically above each other, the arguments of a d-arc horizontally besides each other.

EXAMPLE 1. Below, the formula

$$G = (X \wedge Y) = ((\neg C \wedge A) \vee D \vee E) \wedge (\neg A \vee (B \wedge C)) \tag{1}$$

is displayed as a semantic graph:

$$\begin{array}{c} X\ \boxed{\boxed{\begin{array}{c}\overline{C}\\ \wedge\\ A\end{array}} \vee \boxed{D \vee E}} \\ \wedge \\ Y\ \boxed{\overline{A} \vee \boxed{\begin{array}{c}B\\ \wedge\\ C\end{array}}} \end{array} \tag{2}$$

The boxes in (2) show the explicit subgraphs used in the construction of the semantic graph (since c-arcs and d-arcs are associative and commutative we do not show the explicit subgraphs in subsequent pictorial representations).

DEFINITION 2. If A and B are nodes in a graph, and if $(X, Y)_\alpha$ is an arc ($\alpha = c$ or $\alpha = d$) with A in X and B in Y, we say that $(X, Y)_\alpha$ is the arc *connecting* A and B, and that A and B are *α-connected.*

[2] *true*, *false*, and positive literals represent themselves; a negative literal $\overline{A}$ represents $\neg A$; $(X, Y)_c$ represents $X \wedge Y$; and $(X, Y)_d$ represents $X \vee Y$.

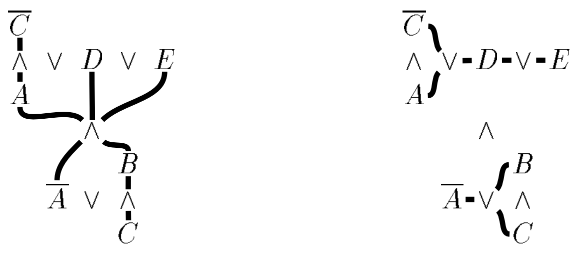
Example 2. In (2), C is c-connected to each of B, A, $\overline{C}$, D, and E, and is d-connected to $\overline{A}$.

Definition 3. 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.

(Partial) d-paths are defined accordingly using d-arcs instead of c-arcs.

$\ell(p)$ denotes the set of literals of a path p.

Example 3. Below, the semantic graph (2) is shown with lines indicating its c-paths (on the left) and its d-paths (on the right):



$\overline{C}$
$\wedge$ $\vee$ D $\vee$ E
A
$\wedge$
B
$\overline{A}$ $\vee$ $\wedge$
C
$\overline{C}$
$\wedge$ $\vee$ D $\vee$ E
A
$\wedge$
B
$\overline{A}$ $\vee$ $\wedge$
C

The c-paths are $\{\overline{C}, A, \overline{A}\}$, $\{\overline{C}, A, B, C\}$, $\{D, \overline{A}\}$, $\{D, B, C\}$, $\{E, \overline{A}\}$, $\{E, B, C\}$; the d-paths are $\{\overline{C}, D, E\}$, $\{A, D, E\}$, $\{\overline{A}, B\}$, $\{\overline{A}, C\}$.

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 of a semantic graph that are not explicit.

Definition 4. Given a semantic graph G and a non-empty subset N of the nodes of G, the *subgraph* of G that corresponds to N is that part of G that consists of nodes from N, where the logical structure of that part is preserved.

$G - N$ denotes the subgraph of G corresponding to the set of nodes of G that are not in N.

Two subgraphs H and H' of G *meet* each other if they have nodes in common.

A non-empty subset N of nodes corresponds unambiguously to one subgraph of G. The empty set corresponds to both *true* and *false*; *true* and *false* are subgraphs of all semantic graphs.

For a more precise definition of subgraphs, see (Murray and Rosenthal, 1993).

EXAMPLE 4. Below the subgraph of (2) is shown that corresponds to the node set $\{A, D, \overline{A}, \}$.

$$\begin{array}{c} A \vee D \\ \wedge \\ \overline{A} \end{array}$$

2.1.2 Blocks

The most important subgraphs are the blocks:

DEFINITION 5. A *c-block* H is a subgraph of a semantic graph G with the property that any c-path p that includes at least one node from H passes through H, where p *passes through* H iff the subset of p consisting of nodes of H is a c-path through H.

d-blocks are accordingly defined using d-paths.

EXAMPLE 5. In (2), the subgraph corresponding 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 sub-path of $\{\overline{A}, C\}$ in the subgraph.

DEFINITION 6. 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.

Full blocks may be treated as essentially explicit subgraphs (up to the order of arguments), and the Isomorphism Theorem from (Murray and Rosenthal, 1987a) assures us that they are the only structures that may be so treated.

EXAMPLE 6. In (2), the subgraph corresponding to $\{\overline{C}, A, E\}$ is a full block. It can be written as $(\{\overline{C}, A\}, E)_d$; i.e., we can regard the upper part of the graph as $(\{\overline{C}, A, E\}, D)_d$. The fundamental subgraphs of the upper disjunction are $(\{\overline{C}\}, \{A\})_c$ and the literals D and E.

DEFINITION 7. 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 use the notation $CE(H)$ and $DE(H)$ for the c- and d-extensions, respectively, of H.

EXAMPLE 7. In (2),

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

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

Path dissolution (Murray and Rosenthal, 1993) is an inferencing mechanism for classical logic that has several interesting properties. It is an efficient generalization of the method of analytic tableaux, is strongly complete in the propositional case, and can produce a list of satisfying interpretations of a formula. The latter feature is particularly valuable in this or in any setting in which one wishes to make use of satisfying interpretations rather than merely to determine whether any exist.

Path dissolution works by selecting a link and restructuring the formula so that all paths through the link are eliminated. The nature of the restructuring is such that one cannot rely on CNF (conjunctive normal form): Even if a formula starts out in CNF, a single dissolution step produces an unnormalized formula. One consequence of eliminating all paths through a link is strong completeness: *Any* sequence of dissolution steps will eventually create a linkless formula. The paths that remain may be interpreted as models (satisfying interpretations) of the formula.

Definition 8. A *c-link* is 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.

Example 8. Consider the link $\{A, \overline{A}\}$ in (2). Then the entire graph $G = (X \wedge Y)$ is the smallest full block containing the link.

Definition 9. Let X be a semantic graph and H an arbitrary subgraph.[3]

The *c-path complement* of H with respect to X, written $CC(H, X)$, is the subgraph of X consisting of all literals in X that lie on c-paths that do not contain nodes from H. If no such literal exists, $CC(H, X) =$ *false*.

The *c-path extension* of H with respect to X, written $CPE(H, X)$, is the subgraph of X containing all literals that lie on c-paths that pass through H. If no such literal exists, $CPE(H, X) =$ *false*.[4]

[3] H usually is but does not have to be a subgraph of X.

[4] Note, that CPE has two arguments whereas CE (Def. 7) has but one; intuitively, CE has an implicit second argument that is always the entire graph in which the explicit argument occurs.

In the development of anti-link operations, we will use 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 are straightforward by duality,

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

EXAMPLE 9. In (2),

$$\begin{aligned} CC(A,X) &= (D \vee E) \\ CPE(A,X) &= (\overline{C} \wedge A) \\ CPE(A,G) &= (\overline{C} \wedge A \wedge Y) \\ CE(A) &= (\overline{C} \wedge A) \end{aligned}$$

The above definitions of the operators CC and CPE are adequate for the definition of dissolution. However, (equivalent) more constructive definitions are given in Section 3, where they will be required in proving the correctness of the anti-link operations introduced there.

The reader is referred to (Murray and Rosenthal, 1993) 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 1. $CPE(H,G)$ 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 2. $CC(H,G)$ 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 $CC(H,G) \vee CPE(H,G)$ and G have the same c-paths.

The above lemmas and corollaries about CC and CPE all hold in dual form for DC and DPE.

Suppose that we have literal occurrences A and $\overline{A}$ residing in conjoined subgraphs X and Y, respectively. It is intuitively clear that the c-paths through $(X \wedge Y)$ that do not contain the link $\{A, \overline{A}\}$ are those through $(CPE(A,X) \wedge CC(\overline{A},Y))$ plus those through $(CC(A,X) \wedge CPE(\overline{A},Y))$ plus those through $(CC(A,X) \wedge CC(\overline{A},Y))$.

DEFINITION 10. Let $H = \{A, \overline{A}\}$ be a link, and let $M = (X, Y)_c$ be the smallest full block containing H. $DV(H, M)$, the *dissolvent of H in M*, is defined 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),

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

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.

The following proposition follows from the corollaries and Lemma 4:

PROPOSITION 1. Either of the two more compact graphs shown below has the same c-paths as $DV(H, M)$, and may thus be used instead:

$$\begin{array}{c} X \\ \wedge \\ CC(\overline{A},Y) \end{array} \vee \begin{array}{c} CC(A,X) \\ \wedge \\ CPE(\overline{A},Y) \end{array} \quad (3) \qquad \begin{array}{c} CC(A,X) \\ \wedge \\ Y \end{array} \vee \begin{array}{c} CPE(A,X) \\ \wedge \\ CC(\overline{A},Y) \end{array} \quad (4)$$

The semantic graphs from the above proposition are not identical to $DV(H, M)$ as graphs, but they 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.

EXAMPLE 10. If we dissolve on the link $\{A, \overline{A}\}$ in (2) (using the compact form (4) of dissolution from Proposition 1), the graph that results is:

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

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 a more general form) can be found in (Murray and Rosenthal, 1993).

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. 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.[5]

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.

DEFINITION 11. A disjunction (conjunction) D *subsumes* another disjunction D' (conjunction D') iff $D \models D'$ ($D' \models D$).
true (*false*) is subsumed by all disjunctions (conjunctions).

A disjunction is called true iff it is equivalent to *true*. A conjunction is called false iff it is equivalent to *false*.

LEMMA 5. If a disjunction (conjunction) D' is not true (false), then D subsumes D' iff $D \subseteq D'$.
A true disjunction (false conjunction) subsumes another true disjunction (false conjunction) only.

As usual, subsumption corresponds to the subset relation, and *only* in the disjunctive case coincides with logical implication.

DEFINITION 12. 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 is a *prime implicate* (*prime implicant*) of a formula G iff

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

Note that the set of all prime implicates (implicants) of a formula G, when treated as a CNF (DNF) formula, is equivalent to G.

DEFINITION 13. Let $\mathcal{D}$ be the set of all prime implicates of a formula G. A prime implicate D of G is *essential* if $\mathcal{D} \setminus \{D\}$ is not equivalent to G, otherwise D is *inessential*.

[5] That means, that the result of applying dissolution repeatedly to an unsatisfiable semantic graph results in the graph *false*, independently of the choice of the link that is dissolved on at each step.

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 set $\ell(p)$ of a path p. In this way, no change in the standard definitions is necessary. The theorem below was proved in (Ramesh and Murray, 1993).

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 3. 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 (Ramesh and Murray, 1993), 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 $\pi(FD(G))$, where

$$\begin{aligned}\pi(F) = \{p \mid\ & p \text{ is a d-path through } F,\ \ell(p) \text{ is not true},\\ & \text{for all d-paths } q \text{ through } F\text{: } \ell(q) \not\subset \ell(p)\}\ .\end{aligned}$$

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

3 Subsumed Paths and Anti-Links

Much of the material in this section is a detailed description of the results sketched in (Beckert *et al.*, 1994). 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.

DEFINITION 14. If $M = (X,Y)_d$ is a d-arc in a semantic graph G and if A_X and A_Y are nodes (occurrences of literal A) in X and in Y respectively, then we call $\{A_X, A_Y\}$ a *disjunctive anti-link.*

If $M = (X,Y)_c$ is a c-arc in G, then we call $\{A_X, A_Y\}$ a *conjunctive anti-link.*

Note, that M is the smallest full block containing the 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 a 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:

(a) 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.

(b) 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'_X must be distinct, or p_Y 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 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

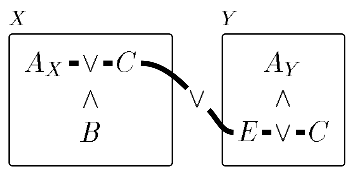
We now identify those disjunctive anti-links which do imply the presence of subsumed paths.

Definition 15. A disjunctive anti-link $\{A_X, A_Y\}$ with respect to the graph G is *redundant* if either $CE(A_X) \neq A$ or $CE(A_Y) \neq A$.

Definition 16. 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.

Then $DP(\{A_X, A_Y\}, G)$ is 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.

Example 11. Consider the following graph $G = (X, Y)_d$:



(5)

The two occurrences of A form a disjunctive anti-link. Because $CE(A_Y) - \{A\} = Y - \{A\}$ and $DPE(A_X, X) = A \vee C$, $DP(\{A_X, A_Y\}, G)$ contains the d-path $p = \{A_X, C, E, C\}$ (indicated by a line). But since $CE(A_X) = A_X$, there are no paths through $CE(A_X) - \{A_X\}$; p is the only member of $DP(\{A_X, A_Y\}, G)$. The anti-link is redundant, and p is subsumed by $p' = \{A_X, C, A_Y\}$ (with literal set $\{A, C\}$). Notice that had G been embedded in a larger graph G', every d-path q containing p in G' would be 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 $\{A_X, A_Y\}$ is an argument of a conjunction, and $DP(\{A_X, A_Y\}, G) \neq \emptyset$. In the above example, the two occurrences of C are both arguments of disjunctions, and thus comprise a non-redundant anti-link for which $DP(\{C_X, C_Y\}, G) = \emptyset$.

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 a semantic graph G. Then each d-path in $DP(\{A_X, A_Y\}, G)$ 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 $p \in DP(\{A_X, A_Y\}, G)$, 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 $(A \wedge C_1 \wedge \ldots \wedge C_n)$, where $n \geq 1$.

Let $p = p_X \cup p_Y \cup p_o$ 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 $p_X = p'_X \cup p_{C_i}$, where p_{C_i} is p_X restricted to C_i, and hence $p = p'_X \cup p_{c_i} \cup p_Y \cup p_o$. The d-path $p'_X \cup \{A\}$ clearly passes through X, and since $A_Y \in p_Y$, $p' = p'_X \cup A_X \cup p_Y \cup p_o$ 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 defined below; in the process, all d-paths in $DP(\{A_X, A_Y\}, G)$ are eliminated, and the two occurrences of the anti-link literal are collapsed into one.

Definition 17. 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

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

Example 12. Consider again the semantic graph (5) from Example 11. We have $DC(A_X, X) = B$ and $DC(A_Y, Y) = (E \vee C)$, so the upper conjunct in $DADV$ is $(B \vee C \vee E)$. For the middle conjunct,

$$\begin{aligned} CE(A_X, X) &= A_X \\ DC(CE(A_X), X) &= B \\ DPE(A_Y, Y) &= A_Y \end{aligned}$$

this conjunct is $(B \vee A)$. Finally in the lower conjunct, $DPE(A_X, X) = (A \vee C)$ and $CC(A_Y, Y) = \mathit{false}$, so this reduces to $(A \vee C)$. The result is:

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

We point out that although $DADV$ produces a CNF formula in the above 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 (Murray and Rosenthal, 1993) and have been defined in Section 2. We present equivalent constructive descriptions here in order to prove Lemma 6 below, and, in the next subsection, to verify that $DADV$ has the desired properties.

Proposition 2. Let G be a semantic graph and H an arbitrary subgraph. Then

$$CPE(H,G) = \begin{cases} \mathit{false} & \text{if } H \text{ does not meet } G \\ G & \text{if } H = G \\ \bigvee_{i=1}^{n} CPE(H_{F_i}, F_i) & \text{if the final arc of } G \text{ is a d-arc} \\ \bigwedge_{i=1}^{k} CPE(H_{F_i}, F_i) \wedge \bigwedge_{j=k+1}^{n} F_j & \text{if the final arc of } G \text{ is a c-arc} \end{cases}$$

$$DPE(H,G) = \begin{cases} \mathit{true} & \text{if } H \text{ does not meet } G \\ G & \text{if } H = G \\ \bigwedge_{i=1}^{n} DPE(H_{F_i}, F_i) & \text{if the final arc of } G \text{ is a c-arc} \\ \bigvee_{i=1}^{k} DPE(H_{F_i}, F_i) \vee \bigvee_{j=k+1}^{n} F_j & \text{if the final arc of } G \text{ is a d-arc} \end{cases}$$

$$CC(H,G) = \begin{cases} G & \text{if } H \text{ does not meet } G \\ \mathit{false} & \text{if } H = G \\ \bigvee_{i=1}^{n} CC(H_{F_i}, F_i) & \text{if the final arc of } G \text{ is a d-arc} \\ \bigwedge_{i=1}^{k} CC(H_{F_i}, F_i) \wedge \bigwedge_{j=k+1}^{n} F_j & \text{if the final arc of } G \text{ is a c-arc} \end{cases}$$

$$DC(H,G) = \begin{cases} G & \text{if } H \text{ does not meet } G \\ \mathit{true} & \text{if } H = G \\ \bigwedge_{i=1}^{n} DC(H_{F_i}, F_i) & \text{if the final arc of } G \text{ is a c-arc} \\ \bigvee_{i=1}^{k} DC(H_{F_i}, F_i) \vee \bigvee_{j=k+1}^{n} F_j & \text{if the final arc of } G \text{ is a d-arc} \end{cases}$$

where F_i $(i \leq i \leq k)$ are the fundamental subgraphs of G that meet H, and F_i $(k+1 \leq i \leq n)$ are those that do not.

LEMMA 6. If G is a graph and A is a literal occurrence in G, then $CC(A, G)$ is logically equivalent to

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

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 result then follows from Lemma 1. The proof is done via induction on the syntactic structure of G (the lemma trivially holds if $G = \mathit{true}$ or $G = \mathit{false}$).

1. If G is a literal, then $G = A$ and both the set of d-paths of $CC(A, G)$ and the set of d-paths of $DPE(A, G) - \{A\}) \wedge DC(CE(A), G)$ are empty. Note, that $DC(CE(A), G) = DC(A, A) = \mathit{true}$, but $(DPE(A, G) - \{A\}) = \{A\} - \{A\} = \mathit{false} = CC(A, A)$.

2. If $G = (X, Y)_d$, 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 $CC(A, G)$ has the same d-paths as $(DPE(A, X) - \{A\}) \wedge DC(CE(A), X) \vee Y$.

   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 semantic graph that has been shown to have the same d-paths as the left hand side.

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

   a) If $CC(A, G) = \mathit{false}$ and thus has no d-paths, then A in X is not d-connected to any other subgraph in X. Hence X is of the form $A \wedge C_1 \wedge \ldots \wedge C_n$ (where $n \geq 0$). But then $G = A \wedge C_1 \wedge \ldots \wedge C_n \wedge Y$, $CE(A) = G$, and $DPE(A, G) = DPE(A, X) = A$. As a result, both $DC(CE(A), G)$ and $DPE(A, G) - \{A\}$ have no d-paths.

   (b) If $CC(A, G) \neq \mathit{false}$, then

   $$CC(A, G) = CC(A, X) \wedge Y \ ,$$

and thus $CC(A,X) \neq \mathit{false}$. Therefore, by the induction hypothesis, $CC(A,G)$ has the same d-paths as

$$(DPE(A,X) - \{A\}) \wedge DC(CE(A),X) \wedge Y \ .$$

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, $CC(A,G) = \mathit{false}$, 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 has the same d-paths as $(DPE(A,X) - \{A\}) \wedge (DC(CE(A),X) \wedge Y)$. 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 the d-paths of $DP(\{A_X, A_Y\}, G)$.

Theorem 6. Let $M = (X,Y)_d$ 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: d-paths in $DP(\{A_X, A_Y\}, G)$ 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 dual of Lemmas 4, X is equivalent to $DC(A_X, X) \wedge DPE(A_X, X)$, and from the distributive law, M is equivalent to

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

Similarly, Y is equivalent to $DC(A_Y,Y) \wedge DPE(A_Y,Y)$, and we expand the upper occurrence of Y and distribute. Thus, M is equivalent to

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

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 d-paths of $DP(\{A_X, A_Y\}, M)$ have been omitted.

Consider the d-paths of $DC(A_X, X)$ — the d-paths in X that miss A_X. They either miss $CE(A_X)$, the c-extension of A_X, or pass through $CE(A_X) - \{A_X\}$. Hence $DC(A_X, X)$ has the same d-paths as

$$DPE(CE(A_X) - \{A_X\}), X) \wedge DC(CE(A_X), X) \ .$$

By replacing the lower occurrence of $DC(A_X, X)$ in the previous graph, we get the following graph M' which is equivalent to M and has the same d-paths as M:

$$M' = \begin{array}{rcl} DC(A_X, X) & \vee & DC(A_Y, Y) \\ & \wedge & \\ \begin{array}{c} DPE((CE(A_X) - \{A_X\}), X) \\ \wedge \\ DC(CE(A_X), X) \end{array} & \vee & DPE(A_Y, Y) \\ & \wedge & \\ DPE(A_X, X) & \vee & Y \end{array}$$

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

$$M'' = \begin{array}{rcl} DC(A_X, X) & \vee & DC(A_Y, Y) \\ & \wedge & \\ DC(CE(A_X), X) & \vee & DPE(A_Y, Y) \\ & \wedge & \\ DPE(A_X, X) & \vee & Y \end{array}$$

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

$$\begin{array}{c} DPE(A_X, Y) \\ \wedge \\ DPE((CE(A_Y) - \{A_Y\}), Y) \\ \wedge \\ DC(CE(A_Y), Y) \end{array}$$

have identical d-paths.

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

$$M''' = \begin{array}{rcl} DC(A_X, X) & \vee & DC(A_Y, Y) \\ & \wedge & \\ DC(CE(A_X), X) & \vee & DPE(A_Y, Y) \\ & \wedge & \\ DPE(A_X, X) & \vee & \begin{array}{c} DPE(A_Y, Y) \\ \wedge \\ DC(CE(A_Y), Y) \end{array} \end{array}$$

The d-paths in M''' are those of M excluding the d-paths in $DP(\{A_X, A_Y\}, M)$. Consider now the d-paths of $DPE(A_X, X) \vee DPE(A_Y, Y)$ 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 $DPE(A_Y, Y)$ to get M''''.

$$M'''' = \begin{array}{rcl} DC(A_X, X) & \vee & DC(A_Y, Y) \\ & \wedge & \\ DC(CE(A_X), X) & \vee & DPE(A_Y, Y) \\ & \wedge & \\ DPE(A_X, X) & \vee & \begin{array}{c} DPE(A_Y, Y) - \{A_Y\} \\ \wedge \\ DC(CE(A_Y), Y) \end{array} \end{array}$$

Applying Lemma 6 to M'''' we get $DADV(\{A_X, A_Y\}, M)$.

In constructing $DADV(\{A_X, A_Y\}, M)$ 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 $DADV(\{A_X, A_Y\}, M)$ is equivalent to M, does not contain the anti-link, and does not contain any d-path of $DP(\{A_X, A_Y\}, M)$. ∎

Theorem 6 gives us a method to remove disjunctive anti-links and some subsumed d-paths: Simply identify a redundant anti-link $H = \{A_X, A_Y\}$ and the smallest full block M containing it, and then replace M by $DADV(H, M)$. 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 $DADV(H, M)$. 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 $DP(\{A_X, A_Y\}, M) \neq \emptyset$) 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, $DADV(\{A_X, A_Y\}, M)$ can be syntactically larger than $M = (X, Y)_d$. 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
$$CE(A_X) = A_X \qquad (\text{and } CE(A_X) \neq X) \ ,$$
then $DC(CE(A_X), X) = DC(A_X, X)$. Therefore by (possibly non atomic) factoring on $DC(A_X, X)$ and observing that $(DC(A_Y, Y) \wedge DPE(A_Y, Y))$ has the same d-paths as Y, $DADV(\{A_X, A_Y\}, M)$ becomes
$$\begin{array}{rcl} DC(A_X,X) & \vee & Y \\ & \wedge & \\ DPE(A_X,X) & \vee & CC(A_Y,Y) \end{array}$$
It turns out that this rule applies to (2) in Example 1. Since $CE(A_X) = A_X$, the simplified rule for this case results in the following graph.
$$\begin{array}{ccc} B & \vee & \begin{array}{c} A \\ \wedge \\ E \vee C \end{array} \\ & \wedge & \\ A & \vee & C \end{array}$$

2. If
$$CE(A_X) = X \ ,$$
then $DC(CE(A_X), X) = \mathit{true}$. Hence
$$DPE(A_X) = A_X \qquad \text{and} \qquad DC(A_X, X) = (X - \{A_X\}) \ .$$
$DADV(\{A_X, A_Y\}, M)$ 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 $CE(A_X, X) = X = A_X$, and the above formula simplifies to

$$A_X \ \lor\ CC(A_Y, Y) \ .$$

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

$$\begin{array}{c} X - \{A_X\} \ \lor\ Y - \{A_Y\} \\ \land \\ A \end{array} \ .$$

This is the graph obtained by disjunctive factoring (Murray and Rosenthal, 1993).

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 $(A_X, Y)_c$. 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 $M = (A_X, Y)_d$. 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. The following definition makes precise the notion of minimality with respect to subsumed d-paths. Then we show that it is NP-hard to achieve this property.

DEFINITION 18. 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 $\ell(p') = \ell(p)$.

The *c-minimal equivalent* of a graph is defined in the obvious dual way.

Note that Property 1 above 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.

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 $\{X_1, \ldots, X_n\}$ be the set of variables in C. Let $A, X_1', \ldots, X_n'$ be distinct variables not occurring in $\{X_1, \ldots, X_n\}$. Let D be the semantic graph obtained by replacing $\overline{X_i}$ by X_i',

$1 \leq i \leq n$, in $\neg C$ (by which we denote the NNF of the negation of C). We construct the following semantic graph G.

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

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 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 D do not contain any $\{X_i, X_i'\}$, $1 \leq i \leq n$; at least one such, say p, is not subsumed by another d-path through 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 D (and hence every d-path through A), there is some i, $1 \leq i \leq n$, 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 d-minimal equivalent and by construction 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. ∎

By a completely dual construction we obtain the following corollary.

COROLLARY 4. 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. Given a conjunctive anti-link $\{A_X, A_Y\}$ in a graph G, determine whether there 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 $\{X_1, \dots, X_n\}$ be the set of variables in C. Let $A, B, X_1', \dots, X_n'$ be distinct variables not occurring in $\{X_1, \dots, X_n\}$. Let D be the semantic graph obtained by replacing $\overline{X_i}$ by X_i', $1 \leq i \leq n$, in $\neg C$. We construct the following semantic graph G.

$$\begin{array}{c} A_1 \vee D \\ \wedge \\ A_2 \vee B \vee \begin{array}{c} X_1 \\ \wedge \\ X_1' \end{array} \vee \dots \vee \begin{array}{c} X_n \\ \wedge \\ X_n' \end{array} \end{array}$$

The subgraph 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 D does not contain any of the literal pairs $\{X_i, X_i'\}$, $1 \leq i \leq n$. 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, $A_2 B p'$ is subsumed by $A_1 p$.

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 no such pair occurs on any d-path in the lower part of G. Therefore p when restricted to D will not have such a literal pair, $\neg C$ has a d-path without a d-link, and hence C is satisfiable. ∎

Corollary 5. The problem dual to the one described in Theorem 9 involving disjunctive anti-links and c-paths is NP-complete.

5 Some Benchmark Examples

Ngair (1993) 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 (Ngair, 1993), 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.

In (Ngair, 1993) a class of formulas is proposed 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 in this case the subsumed implicates correspond to easily recognizable anti-links of both the disjunctive and conjunctive kind. We show that if these anti-links are removed after dissolution is performed, dissolution + Pi can find all the implicates in polynomial time.

Ngair's formulas are abbreviated with F_n $(n \geq 1)$ and are defined as:

$$F_n = \left(\left(\bigwedge_{i=2}^{n} \overline{A_{2i-1}} \right) \vee A_1 \right) \wedge \left(\left(\bigwedge_{i=2}^{n} \overline{A_{2i}} \right) \vee A_2 \right) \wedge \bigvee_{i=1}^{n} (A_{2i-1} \wedge A_{2i})$$

In the left part of Figure 1 we show the graph of F_n for a fixed n.

Clearly F_n has $4n$ literals, and $2n-2$ c-links; dissolution can remove these links by performing $2n$ dissolution steps. The full dissolvent that results is depicted in the right part of Figure 1.

The structure of the full dissolvent depends on the order in which links are selected for application of dissolution; the above dissolvent is the one obtained by the current version of our propositional dissolution prover Dissolver. (The compact version (3) from Proposition 1 of the dissolvent 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 in Figure 2. (Since Dissolver is NNF-based, such factoring is not only feasible but is in fact implemented and routinely employed to produce the final output.)

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:

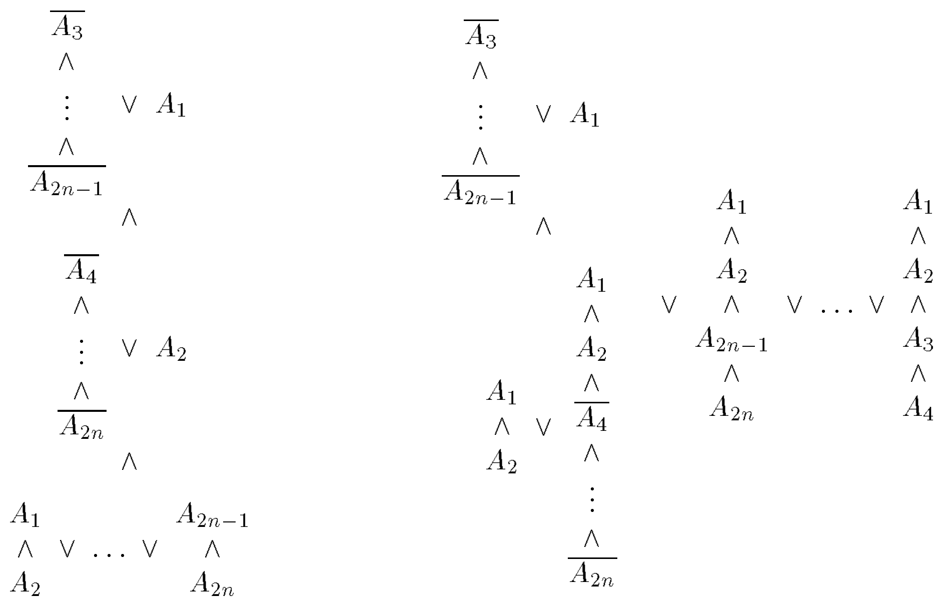
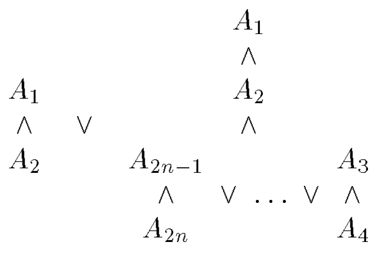


Fig. 1. Semantic graph of Ngair's formulas before and after dissolution.



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.

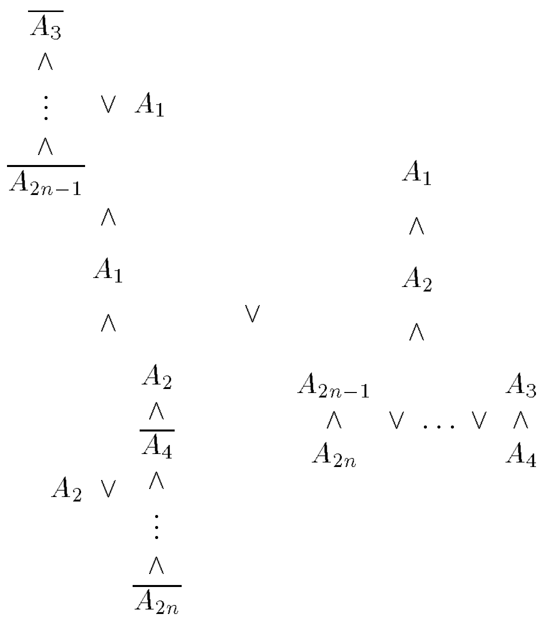


Fig. 2. Ngair's formulas after dissolution and factoring.

6 A Generalized Purity Principle

6.1 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.

Definition 19. A subgraph M in a graph G is *pure* iff all c-links or d-links that meet M at all are totally within M.[6] If, in addition, all conjunctive or disjunctive anti-links that meet M at all are totally within M, we say that M is *strictly pure*.[7] If M is a full block in G we speak of a *(strictly) pure full block*.

When factored, some of the examples from (Ngair, 1993) 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

[6] This is just the obvious generalization of the concept of a pure literal as it is used in the literature on CNF-based automated deduction.

[7] Simply put, M shares no variables with the rest of G.

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.

The following theorems characterize the properties of strictly pure full blocks with respect to prime implicates. In them we employ the following notation: let M be an explicit subgraph of a graph G and let X be a variable not occurring in G. By G^X_M we denote the graph obtained by the substitution of X for M in G. Similarly, if D is a disjunction of literal occurrences from G we denote by D_M the disjunction of literals that occur in M and by D_{G-M} the disjunction of literals that do not. Obviously, $D = (D_{G-M} \vee D_M)$ holds. Finally, we set

$$D^X_M = \begin{cases} D_{G-M} \vee X & \text{if } D_M \neq \textit{false} \text{ (empty disjunction)} \\ D_{G-M} & \text{otherwise} \end{cases}$$

THEOREM 10. Let M be a satisfiable strictly pure full block in a satisfiable semantic graph G and let D be a non-tautological disjunction of literals from G. If $D_M \neq$ *false*, then the following statements are equivalent:

1. D is a prime implicate of G.

2. D^X_M is a prime implicate of G^X_M, and D_M is a prime implicate of M.

This theorem turns out to be a special case of Theorem 11 to be proved in the following subsection.

If a graph G contains several strictly pure full blocks $M_1, \ldots, M_n$, then the repeated application of Theorem 10 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 $(1 \leq i \leq n)$ and compute the prime implicates of the resulting graph G^X_M. Then, all substitutions of prime implicates of M_i for X_i in the prime implicates of G^X_M 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^X_M that can be extended through M_i to form a d-path in G. Observe that prime implicates of G^X_M containing none of the variables X_i $(1 \leq i \leq n)$ are simply prime implicates of G that do not contain literals from any of the blocks M_i.

Several *distinct* strictly pure full blocks can be handled by repeated application of Theorem 10 as explained above. However, multiple occurrences $M^1, \ldots, M^n$ of a *single* full block M in G require an extended 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 occurrences M^i in G by the single new variable X, the prime implicates of the resulting graph G^X_M would also bear a strong relationship to those of G. This is made precise in the next section.

6.2 MULTI-PURE FULL BLOCKS

DEFINITION 20. Suppose that $M^1, \dots, M^n$ are occurrences of full blocks in G and that all of them are syntactically identical (up to associativity and commutativity of disjuncts and conjuncts). The subgraph M^* formed by taking all the nodes of the blocks M^i is not necessarily a full block;[8] but let M^* be strictly pure. Then we call the M^i *multi-pure full blocks.*

In addition, suppose that there are occurrences $\overline{M^{n+1}}, \overline{M^{n+2}}, \dots, \overline{M^{n+m}}$ of full blocks syntactically identical (up to associativity and commutativity of disjuncts and conjuncts) to $\overline{M}$, the NNF of the complement of any of the M^i $(1 \le i \le n)$. We call $M^1, \dots, M^n, \overline{M^{n+1}}, \overline{M^{n+2}}, \dots, \overline{M^{n+m}}$ *complementary multi-pure full blocks.*

Note that each (complementary) multi-pure full block M^i is *not* strictly pure, since it has anti-links (and possibly links) to its equivalent (complementary) full blocks in G.

Observe that complementary non-atomic formulas must be recognized. For example, if $M = (A \vee B)$, then $\overline{M}$ could be $\overline{A} \wedge \overline{B}$ or $\overline{B} \wedge \overline{A}$. In fact, $\overline{M}$ could have been input as $\neg(A \vee B)$ or as $\neg(\neg A \supset B)$. If NNF formulas are stored in an appropriate canonical way, complementarity is easily (that is, in polynomial time) detectable; the situation is also straightforward when complementary formulas have the form M and $\neg M$ prior to conversion to NNF. In any case, a detailed treatment of this issue is beyond the scope of this paper. We do note that in the absence of complements, multi-pure full blocks are recognizable in polynomial time via a canonical NNF representation. A modification of any algorithm for finding common subtrees (see (Grossi, 1993) for one such algorithm) can be used for recognizing multi-pure full blocks.

It turns out that the results of Theorem 10 can be extended to the case in which a formula contains complementary multi-pure full blocks. We use the notation of Theorem 10 with the understanding that M denotes any occurrence of an $M^1, M^2, \dots, M^n$ and the $\overline{M^{n+1}}, \overline{M^{n+2}}, \dots, \overline{M^{n+m}}$ are treated as negated occurrences of M (thus G^X_M replaces the complementary occurrences $\overline{M^j}$ as well). M^* is defined as the subgraph of G relative to the M^i and $\overline{M}^j$ $(1 \le i \le n < j \le n+m)$. G^X_M, D_M, D_{G-M}, and D^X_M are defined as before, but relative to M^*. Additionally, we define $D^{\overline{X}}_M$ in the obvious way (Intuitively, we use D^X_M when the literals of D_M correspond to unnegated occurrences of M, and we use $D^{\overline{X}}_M$ when the literals in D_M correspond to negated occurrences of M).

THEOREM 11. Let $M^1, M^2, \dots, M^n$ and $\overline{M^{n+1}}, \overline{M^{n+2}}, \dots, \overline{M^{n+m}}$ be complementary multi-pure full blocks in a satisfiable semantic graph G, where all of the blocks M^i and $\overline{M^j}$ are satisfiable (we allow $m = 0$). Let D be a non-tautological disjunction of literals from G. Then the following statements are equivalent:

[8] As the blocks M^i could be single literals occurring in arbitrary positions, this is hardly surprising.

1. D is a prime implicate of G.

2. $D_M = \mathit{false}$ and D is a prime implicate of G_X

   or

   $D_M \neq \mathit{false}$ and D_M^X is a prime implicate of G_M^X, and D_M is a prime implicate of M

   or

   $D_M \neq \mathit{false}$ and $D_M^{\overline{X}}$ is a prime implicate of G_M^X, and D_M is a prime implicate of $\overline{M}$.

Proof. Let ${G'}_M^X$ be a graph without c-links that is equivalent to G_M^X (for instance, ${G'}_M^X$ could be the full dissolvent of G_M^X). Similarly, let M' be a c-linkless equivalent of M, and $\overline{M}\,'$ be a c-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_M^X is present as a d-path in ${G'}_M^X$.

To prove the only-if-part, let D be a prime implicate of G. Then there must be 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 occurrences of M or $\overline{M}$. In this case, p must also be a d-path in ${G'}_M^X$ (technically, p is isomorphic to a d-path in ${G'}_M^X$). To prove that p is not subsumed by another d-path in ${G'}_M^X$, assume otherwise, namely, that there is a path p' in ${G'}_M^X$ that 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 ${G'}_M^X$ and hence of G_M^X.

Now suppose p passes through the full blocks ${M^{i_1}}', \ldots, {M^{i_q}}'$. Let p_j, $1 \leq j \leq q$, be the restriction of p to ${M^{i_j}}'$. Notice that for $1 \leq j,\, k \leq q$, $\ell(p_j) = \ell(p_k)$; 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 ${M^{i_j}}'$ which is 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 by X in p; p^X is a d-path in G_M^X. Furthermore, p^X cannot be subsumed by another d-path in G_M^X (again, such a subsuming path would induce a path in G' that subsumes p). Therefore $D_M^X = \ell(p^X)$ is a prime implicate of G_M^X.

Finally, in the case that p passes through the full blocks ${\overline{M}^{i_1}}', \ldots, {\overline{M}^{i_r}}'$, the argument is similar as in the previous paragraph.

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

Now suppose that D_M^X is a prime implicate of G_M^X and that D_M is a prime implicate of M. Then there are unsubsumed d-paths p^X in G'^X_M and p_M in M', respectively, such that $\ell(p^X) = D_M^X$ and $\ell(p_M) = D_M$, respectively. In the present subcase, by definition, D_M^X contains X (and cannot contain $\overline{X}$). 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_M^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 $D_M^{\overline{X}}$ is a prime implicate of G_M^X and D_M is a prime implicate of $\overline{M}$, then D is a prime implicate of G. ∎

It is straightforward to see that in the case when $n = 1$ and $m = 0$ Theorem 11 collapses into Theorem 10.

On the one hand we expect to substitute new variables for 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 as it is defined in (Murray and Rosenthal, 1993)).

That dissolving on two complementary full blocks accomplishes exactly what dissolving on all the corresponding single-links 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 desirable. 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 condition 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.

6.3 More Examples

Kean & Tsiknis (1990) provide a class of examples referred in the following to as K_{nm}. They have $mn+1$ input CNF clauses and $(m+1)^n+mn$ prime implicates. This set of clauses can be factored to obtain a more compact representation in NNF as shown in Figure 3.

Since the number of prime implicates is exponential, so is the number of subsumption checks required. The number of subsumption checks for the Ipia (de Kleer, 1992) and Gen-Pi (Ngair, 1993) algorithms are shown in Table I.

For each i, the literals $\overline{S_{i1}}, \ldots, \overline{S_{im}}$ 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

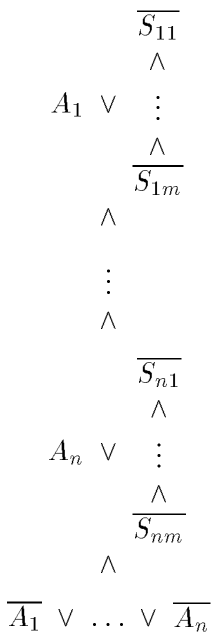


Fig. 3. Semantic graph of Kean & Tsiknis's formulas.

Examples	IPIA	GEN-PI	PI + anti-link
K_{33}	5166	972	164
K_{44}	506472	11600	887
K_{54}	1730120	29074	887

TABLE I
Number of subsumption checks needed for K_{mn} by IPIA, GEN-PI, and PI + anti-link.

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 I, 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) and of the general case of K'_{mn} $(n > 2)$ are shown in Figure 4.

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

Fig. 4. The full dissolvent of K_{m2} (left) and of K_{mn}, $n > 2$ (right).

The number of subsumption checks required here is also shown in Table I. Clearly, our techniques produce a significant reduction in the number of subsumption checks required. Note that for the problem K_{mn}, the number of subsumption checks depends only on n and not on m, and is not reduced by applying the anti-link operations to the full dissolvent.

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 $\overline{S_{i1}}, \ldots, \overline{S_{im}}$ in the unfactored form of K_{mn} do not form a 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 (Ramesh and Murray, 1995). 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.)

7 Comparison with BDDs

BDDs (Bryant, 1986) are commonly used in verification of boolean circuits. Coudert and Madre (Coudert and Madre, 1993) describe an algorithm which produces the prime implicates/implicants of a propositional formula represented as a BDD. Any algorithm including theirs which uses BDDs must perform large amounts of subsumption testing. Given any formula in NNF, a BDD based method would first construct the BDD and then extract the prime implicates/implicants from it. In contrast, our system would first compute the full dissolvent. But for either of these approaches, the next stage — extracting the prime implicates/implicants — requires extensive testing for subsumption.

The size of the BDD depends critically on the ordering (of variables) chosen. There are classes of NNF formulas (Breitbart *et al.*, 1995) for which any BDD will be exponentially large in the formula size. These formulas do not have any c-links, so the dissolution phase of our method does not change the formula. Hence the input to PI would be a small formula, whereas the BDD based method would have to handle an exponentially larger intermediate representation. However these formulas have many prime implicates and prime implicants, and the subsumption checking is the bottleneck for both methods.

In all likelyhood, the relative performance of these two methods can be determined through experimental evaluation only; formulas will exist for which one method is superior, and vice versa.

8 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 (Ngair, 1993).

Some experimental results on a dissolution- and PI-based system for computing prime implicates are reported in (Ramesh and Murray, 1993). That system is currently being extended; some anti-link operations are already implemented and have shown to improve performance. Operations based on strictly pure full blocks are under development, and their effectiveness in practice will be tested.

References

Bernhard Beckert, Reiner Hähnle, Anavai Ramesh, and Neil V. Murray. On anti-links. In F. Pfenning, editor, *Proceedings, 5th International Conference on Logic Programming and Automated Reasoning (LPAR), Kiev, Ukraine*, LNCS 822, pages 275–289. Springer, 1994.

Yuri Breitbart, Harry B. Hunt, and Daniel Rosenkrantz. On the size of binary decision diagrams representing boolean functions. *Theoretical Computer Science*, 145(1–2):45–70, July 1995.

Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE trans. on computers*, c-35(8):677–691, August 1986.

Olivier Coudert and Jean-Christophe Madre. A new graph based prime computation technique. In T. Sasao, editor, *Logic Synthesis and Optimization*, chapter 2, pages 33–58. Kluwer, Norwell/MA, USA, 1993.

Johan de Kleer and B. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

Johan de Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2–3):197–222, August 1992.

Johan de Kleer. An improved incremental algorithm for computing prime implicants. In *Proceedings, AAAI-92, San Jose, CA*, pages 780–785, 1992.

Roberto Grossi. On finding common subtrees. *Theoretical Computer Science*, 108(2):345–356, 1993.

Peter Jackson and John Pais. Computing prime implicants. In *Proceedings, 10th International Conference on Automated Deduction (CADE), Kaiserslautern, Germany*, LNCS 449, pages 543–557. Springer, July 1990.

Peter Jackson. Computing prime implicants incrementally. In *Proceedings, 11th International Conference on Automated Deduction (CADE), Saratoga Springs, NY*, LNCS 607, pages 253–267, June 1992.

Alex Kean and George Tsiknis. An incremental method for generating prime implicants/implicates. *Journal of Symbolic Computation*, 9:185–206, 1990.

Alex Kean and George Tsiknis. Assumption based reasoning and clause management systems. *Computational Intelligence*, 8(1):1–24, November 1992.

Reinhold Letz, Johann Schumann, Stephan Bayerl, and Wolfgang Bibel. SETHEO: A high-performance theorem prover. *Journal of Automated Reasoning*, 8(2):183–212, 1992.

Igor Mozetič and Christian Holzbaur. Controlling the complexity in model-based diagnosis. *Annals of Mathematics and Artificial Intelligence*, 11:297–314, 1994.

Neil V. Murray and Eric Rosenthal. Inference with path resolution and semantic graphs. *Journal of the ACM*, 34(2):225–254, April 1987.

Neil V. Murray and Eric Rosenthal. Path dissolution: A strongly complete rule of inference. In *Proceedings, 6th National Conference on Artificial Intelligence, Seattle, WA*, pages 161–166, July 1987.

Neil V. Murray and Eric Rosenthal. Dissolution: Making paths vanish. *Journal of the ACM*, 48(3):504–535, July 1993.

Teow-Hin Ngair. A new algorithm for incremental prime implicate generation. In *Proceedings, IJCAI-93, Chambery, France*, August 1993.

Teodor C. Przymusinski. An algorithm to compute circumscription. *Artificial Intelligence*, 30:49–73, 1989.

Anavai Ramesh and Neil V. Murray. Non-clausal deductive techniques for computing prime implicants and prime implicates. In *Proceedings, 4th International Conference on Logic Programming and Automated Reasoning (LPAR), St. Petersburg, Russia*, LNCS 698, pages 277–288. Springer, July 1993.

Aanavai Ramesh and Neil Murray. An application of non clausal deduction in diagnosis. In *Proceedings, Eighth International Symposium on Artificial Intelligence, Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey, Mexico*, pages 378–385, October 1995.

Tsutomu Sasao. Logic synthesis with EXOR gates. In T. Sasao, editor, *Logic Synthesis and Optimization*, chapter 12, pages 259–286. Kluwer, Norwell/MA, USA, 1993.

James R. Slagle, Chin-Liang Chang, and Richard C. T. Lee. A new algorithm for generating prime implicants. *IEEE Transactions on Computers*, C-19(4):304–310, 1970.