Mathware & Soft Computing 4 (1997) 155-179

Parameterized Prime Implicant/Implicate Computations for Regular Logics*

Anavai Ramesh[1]† and Neil V. Murray
[1] Intel Corporation. Chandler, Arizona
agramesh@sedona.intel.com
[2] Dept. of Computer Science. SUNY at Albany, NY.
nvm@cs.albany.edu

Abstract

Prime implicant/implicate generating algorithms for multiple-valued logics (MVL's) are introduced. Techniques from classical logic not requiring large normal forms or truth tables are adapted to certain "regular" multiple-valued logics. This is accomplished by means of signed formulas, a meta-logic for multiple valued logics; the formulas are normalized in a way analogous to negation normal form. The logic of signed formulas is classical in nature.

The presented method is based on path dissolution, a strongly complete inference rule. The generalization of dissolution that accommodates signed formulas is described. The method is first characterized as a procedure iterated over the truth value domain $\Delta = \{0, 1, \ldots, n-1\}$ of the MVL. The computational requirements are then reduced via parameterization with respect to the elements and the cardinality of Δ.

1 Introduction

Prime implicant generating algorithms are used as the first step in minimization of functions in classical two-valued logic [18] and in multiple-valued logics (MVL's) [1, 17, 19]. In artificial intelligence research, prime implicate generating algorithms have been used in clause management and in truth maintenance systems [8, 15]. Many algorithms for computing prime implicants and prime implicates have been developed; for example, Slagle, Chang, and Lee [16], Kean and Tsiknis [7], Jackson and Pais [5], Jackson [6], and Strzemecki [18] for classical logic, Su and Cheung [19], Smith III [17] and Allen and Givone [1] for multiple-valued logics. These algorithms require the input to be in some normal form — usually conjunctive normal form (CNF), disjunctive normal form (DNF), or as a set of

*This research was supported in part by National Science Foundation Grants CCR-9101208 and CCR-9404338.

†This work was carried out while the author stayed at the University at Albany.

minterms. Ngair's method [13] employs a normal form which is much more restrictive than negation normal form (NNF) although less restrictive than CNF or than DNF.

In [14] we gave several related methods for finding prime implicants/implicates when the input is in the less restrictive NNF. One can easily convert boolean functions containing negation, conjunction, implication, and disjunction to NNF with no increase in formula size. We showed theoretically as well as empirically that by working with formulas in NNF, running times are improved in comparison with those obtainable on equivalent CNF or DNF inputs. Our methods are based on dissolution [9], a strongly complete inference rule, and an algorithm called PI. (By strongly complete, we mean that *any* sequence of dissolution steps will terminate in a linkless formula equivalent to the original — see [9].) The path-based techniques from [5] are combined with techniques that are inference-based as in [6, 7]. A key feature of our techniques is that they do not rely on CNF, DNF, or on minterms.

In this paper we investigate the adaptability of our techniques to multiple-valued logics. Dissolution is a sound and complete inference rule; it was first proposed for classical two-valued logic and handles formulas in NNF. It was later generalized to handle multiple valued logics through *signed formulas*, a classical logic that serves as a meta-logic for multiple-valued logics. Signed formulas (defined below) capture, at a meta-level, queries about formulas of multiple-valued logics. The answers to these queries are either *true* or *false*. For example, let $\mathcal{F}$ be a formula in some 4 valued logic with {0,1,2,3} as the set of truth values. To answer queries such as "can the value of $\mathcal{F}$ be 1" or "can the value of $\mathcal{F}$ be greater than 1," we express them as the signed formulas $\{1\}{:}\mathcal{F}$ and $\{>1\} : \mathcal{F}$, respectively. The answer to such queries is yes if the corresponding signed formula is satisfiable, *false* if not. Satisfiability for signed formulas is defined in the classical way, but under certain restrictions allowing only those assignments that correspond to "reasonable" assignments over the multiple-valued language. Depending upon the deductive machinery employed, affirmative answers to such queries may be accompanied by a representation of those interpretations under which the query comes out *true*. It is this feature that is crucial for producing prime implicants/implicates.

We first define the notion of *prime s-implicant* and *prime s-implicate* with respect to signed formulas that are normalized in a way analogous to NNF. These are generalizations to signed formulas of the classical notions of prime implicant/implicate. Then, to find the prime implicants/implicates of multiple-valued logic formulas, we first form the corresponding signed formulas, find the prime s-implicants/s-implicates of these signed formulas, and finally translate them to prime implicants/implicates of the multiple-valued function.

The method is first characterized as a procedure iterated over the truth value domain $\Delta = \{0, 1, \ldots, n-1\}$ of the MVL. The computational requirements are then reduced via parameterization with respect to the elements and the cardinality of Δ. Our method therefore provides a way of finding prime implicants or prime implicates that is quite independent of the particular MVL employed (within the class of MVL's called *regular* logics, defined in Section 2.1).

In the next section we introduce basic definitions including signed formulas and path dissolution. The notions of implicant and implicate are generalized to

s-implicant and s-implicate for signed formulas in Section 3. In Section 4 we adapt our classical methods to produce a new algorithm for finding the prime s-implicants of an MVL formula based on signed formulas. The algorithm's correctness is shown. In Section 5 we show how our approach yields a method for computing the prime implicants of formulas in a family of Post logics.

2 Signed Formulas

Signed formulas have been used by Hähnle [3] and by Murray and Rosenthal [10, 11] to build inference systems for multiple valued logics. A review of signed formulas and of the corresponding propositional logic appears below; it is borrowed from [11] in order to make this paper self-contained.

We assume a (propositional) language Λ consisting of logical formulas built in the usual way from a set $\mathcal{A}$ of atoms, a set Γ of connectives, and a set χ of logical constants. For the sake of completeness, we precisely define a *formula* in Λ as follows:

1. Atoms and logical constants are formulas.
2. If Θ is a connective of arity n and if $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n$ are formulas, then so is $\Theta(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_n)$.

Associated with Λ is a set Δ of truth values, and an interpretation for Λ is a function from $\mathcal{A}$ to Δ; i.e., an assignment of truth values to every atom in Λ. A connective Θ of arity n denotes a function $\Theta : \Delta^n \longrightarrow \Delta$. Interpretations are extended in the usual way to mappings from formulas to Δ.

We use the term *sign* for any subset of Δ (and overload it by also using it for any expression that denotes a subset of Δ). We define a *signed formula* to be an expression of the form S:$\mathcal{F}$, where S is a sign and $\mathcal{F}$ is a formula in Λ; if $\mathcal{F}$ is an atom in Λ, we call S:$\mathcal{F}$ a *signed literal*. We are interested in signed formulas because they represent queries of the form, "Are there interpretations under which $\mathcal{F}$ evaluates to a truth value in S?" To answer arbitrary queries, we map formulas in Λ to formulas in a classical propositional logic Λ_s, called *signed formulas*; it is defined as follows: The atoms are signed formulas and the connectives are (classical) conjunction and disjunction. The set of truth values is of course $\{true, false\}$.

An arbitrary interpretation for Λ_s may make an assignment of *true* or *false* to any signed formula (i.e., to any atom) in the usual way. Our goal is to focus attention only on those interpretations that relate to the sign in a signed formula. To accomplish this we restrict attention to Λ-*consistent interpretations*. An interpretation I over Λ assigns to each atom, and therefore to each formula $\mathcal{F}$, a truth value in Δ, and the corresponding Λ-consistent interpretation I_S is defined by $I_S(S : \mathcal{F}) = true$ if $I(\mathrm{F}) \in S$, and $I_S(S : \mathcal{F}) = false$ if $I(\mathcal{F}) \notin S$. Note that this is a 1–1 correspondence between the set of all interpretations over Λ and the set of Λ-consistent interpretations over Λ_S. Intuitively, Λ-consistent means an assignment of *true* to all signed formulas whose signs are simultaneously achievable via some interpretation over the original language. Restricting attention to

Λ-consistent interpretations yields a new consequence relation: If $\mathcal{F}_1$ and $\mathcal{F}_2$ are formulas in Λ_s, we write $\mathcal{F}_1 \models_\Lambda \mathcal{F}_2$ if whenever I_s is a Λ-consistent interpretation and $I_s(\mathcal{F}_1) = true$, then $I_s(\mathcal{F}_2) = true$.

The following lemma is immediate.

Lemma 1 Let I_s be a Λ-consistent interpretation, let A be an atom and $\mathcal{F}$ a formula in Λ, and let S_1 and S_2 be signs. Then:

i) $I_s(\emptyset : \mathcal{F}) = false$;

ii) $I_s(\Delta : \mathcal{F}) = true$;

iii) $S_1 \subseteq S_2 \Leftrightarrow S_1 : \mathcal{F} \models_\Lambda S_2 : \mathcal{F}$, for all formulas $\mathcal{F}$;

iv) There is exactly one $\delta \in \Delta$ such that $I_s(\{\delta\} : A) = true$. □

The next lemma follows immediately from part iv of Lemma 1. First, we say that two formulas $\mathcal{F}_s$ and $\mathcal{F}'_s$ in Λ_s are Λ-*equivalent* if $I_s(\mathcal{F}_s) = I_s(\mathcal{F}'_s)$ for any Λ-consistent interpretation I_s; we write $\mathcal{F}_s \equiv_\Lambda \mathcal{F}'_s$. Observe that $\mathcal{F}_s \equiv_\Lambda \mathcal{F}'_s$ if and only if $\mathcal{F}_s \models_\Lambda \mathcal{F}'_s$ and $\mathcal{F}'_s \models_\Lambda \mathcal{F}_s$.

Lemma 2 (*The Reduction Lemma*). Let $S_1 : A$ and $S_2 : A$ be signed literals in Λ_s; then

$$S_1 : A \wedge S_2 : A \equiv_\Lambda (S_1 \cap S_2) : A \qquad \textit{and}$$

$$S_1 : A \vee S_2 : A \equiv_\Lambda (S_1 \cup S_2) : A$$

□

2.1 Λ-atomic formulas

We say that a formula in Λ_s is Λ-*atomic* if for each signed literal $S : A$, A is an atom in Λ; i.e., if the only atoms in the formula are signed literals. In [11], it was shown that a Λ-equivalent version of any signed formula can in principle be computed when Δ is finite. In general, however, computing the Λ-atomic equivalent of $S : \mathcal{F}$ can be prohibitively expensive. Yet for many applications in which prime implicants/implicates are useful (e.g., hardware design), the logic employed is highly structured and turns out to be a *regular logic* as defined by Hähnle [2]. Intuitively, a regular logic has a finite linearly ordered truth domain, and its operators behave monotonically with respect to each of their arguments; for details, see [2].

For example, the family of Post's Logics P_n using $\vee_p$, $\wedge_p$, *and* σ as connectives can be expressed as regular logics. Here $\sigma(A) = (A+1) \bmod n$ and $\vee_p$ and $\wedge_p$ are defined as **max** and **min**. Although σ is not regular, it can be defined in terms of regular unary operators — see Section 5.

Consider the following signed formula in P_4, $\{> 1\} : \mathcal{F}$ where $\mathcal{F}$ is $((x_1 \vee_p x_2) \wedge_p \sigma(x_3))$. After driving the signs inward we get the following (Λ-equivalent — see Section 5) Λ-atomic formula $\mathcal{F}' = ((\{> 1\} : x_1 \vee \{> 1\} : x_2) \wedge (\{> 1\} :$

$$((\neg S_1 : C \wedge S_2 : A) \vee S_3 : C) \wedge (\neg S_4 : A \vee (S_5 : B \wedge S_6 : C)) \equiv$$

$$\begin{array}{ccc} S_1 : C & & \\ \wedge & \vee & S_3 : C \\ S_2 : A & & \end{array} \qquad \begin{array}{ccc} & \wedge & \\ & & S_5 : B \\ S_4 : A & \vee & \wedge \\ & & S_6 : C \end{array}$$

Figure 1: NNF in two dimensions

$x_3 \wedge \{< 3\} : x_3)$). Note that the operators $\vee$ and $\wedge$ in $\mathcal{F}'$ are classical disjunction and conjunction, not **max** and **min** over Δ.

Let $\mathcal{F}$ be any signed formula. We introduce classical negation ($\neg$) into signed formulas by defining $\neg\mathcal{F}$ as follows: If I_s is a Λ-consistent interpretation, $I_s(\neg\mathcal{F}) = \mathit{true}$ if $I_s(\mathcal{F}) = \mathit{false}$; otherwise, $I_s(\neg\mathcal{F}) = \mathit{false}$. The proofs of the identities listed below follow from Λ-consistency and well known properties of classical two-valued logic.

1. $\neg S : \mathcal{F} \equiv_\Lambda \overline{S} : \mathcal{F}$, where $\overline{S} = \Delta - S$.
2. $\neg(S_1 : \mathcal{F}_1 \wedge S_2 : \mathcal{F}_2) \equiv_\Lambda \neg S_1 : \mathcal{F}_1 \vee \neg S_2 : \mathcal{F}_2$.
3. $\neg(S_1 : \mathcal{F}_1 \vee S_2 : \mathcal{F}_2) \equiv_\Lambda \neg S_1 : \mathcal{F}_1 \wedge \neg S_2 : \mathcal{F}_2$.
4. $S_1 : \mathcal{F}_1 \models_\Lambda S_2 : \mathcal{F}\ iff\ \neg S_1 : \mathcal{F}_1 \vee S_2 : \mathcal{F}_2$ is valid w.r.t. Λ-consistency.

2.2 NNF notation

A number of technical terms and definitions are introduced in this section. Although the treatment is brief, the examples plus the reader's intuition should provide an adequate understanding of this background material. For a more detailed exposition, see [9].

The formulas in Λ_s that we are interested in are in *negation normal form* (NNF): The only connectives used are conjunction and disjunction. Such NNF formulas naturally lend themselves to a two-dimensional representation that we call *semantic graphs*. A *semantic graph* G is either one of the truth constants *true* or *false*, a literal, a c-arc which is a conjunction of two semantic graphs, or 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. For example, in Figure 1, the formula on the left is displayed graphically on the right.

Disjunctions (d-arcs) are displayed horizontally, conjunctions (c-arcs) vertically. 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. The graph above contains four *c-paths* (maximal conjunctions

Figure 2: Subgraphs

of literal occurrences): $\{S_1 : C, S_2 : A, S_4 : A\}$, $\{S_3 : C, S_5 : B, S_6 : C\}$, $\{S_1 : C, S_2 : A, S_5 : B, S_6 : C\}$, $\{S_3 : C, S_4 : A\}$.

More precisely, if A and B are nodes in a graph, and if $\mathbf{a} = (X, Y)_\alpha$ is an arc ($\alpha = c$ or $\alpha = d$) with A in X and B in Y, we say that $\mathbf{a}$ is the arc *connecting* A and B, and that A and B are *α-connected*. In Figure 1, $S_1 : C$ is c-connected to each of $S_2 : A$, $S_4 : A$, $S_5 : B$, and $S_6 : C$, and is d-connected to $S_3 : C$. 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; similarly for *d-paths*.

We define a *c-link* to be a minimal set of mutually unsatisfiable pairwise c-connected signed literals and a *d-link* to be a minimal set of mutually valid pairwise d-connected signed literals. Some thought should convince the reader that all link elements share the same atom from Λ and that either the intersection of their signs is $\emptyset$ (for c-links) or the union is Δ (for d-links). Note that for the special case of classical logic, this reduces to the usual definition of a pair of complementary literals. In Λ_S, however, links are not necessarily binary. Since dissolution was designed to operate on pairs of objects, we define a *partial c-link* to be a pair of c-connected literals that differ only in their signs; similarly for *partial d-links*. For example in the above figure $S_2 : A$ and $S_4 : A$ form a partial c-link, and if $S_2 \cap S_4 = \emptyset$, then they also form a c-link.

A semantic graph is unsatisfiable (valid) if and only if every c-path (d-path) is unsatisfiable (valid), and a c-path (d-path) is unsatisfiable (valid) if and only if it contains a c-link (d-link).

It is 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. We use G_X to denote the subgraph of G with respect to a set of nodes X. The example of Figure 1 is shown on the left in Figure 2; the subgraph relative to the set $\{S_2 : A,\ S_4 : A,\ S_6 : C\}$ is the graph on the right.

Note that every c-path (d-path) in G_X will be a partial c-path (d-path) in G. A non-empty subset N of nodes corresponds unambiguously to one subgraph of G. The empty set corresponds to both *true* and to *false*; *true* and *false* are subgraphs of all semantic graphs. For a precise definition of subgraphs, see [9].

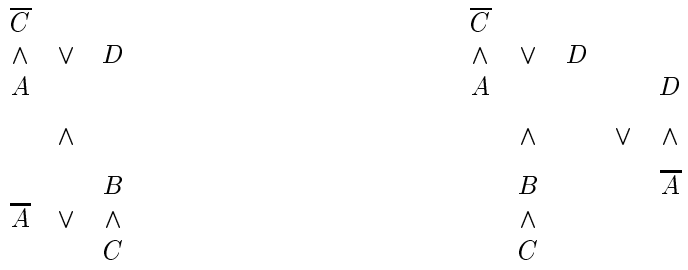

Figure 3: Classical dissolution

2.3 Path dissolution

In classical logic, the idea of path dissolution is to eliminate all paths through a given c-link. Given a formula G, repeated applications of dissolution will eventually produce a linkless formula $FD(G)$ called the *full dissolvent* of G. Hence all remaining paths are satisfiable, so G is unsatisfiable if and only if $FD(G)$ has no c-paths; i.e., if and only if $FD(G)$ is the empty disjunction.

We can get an intuitive idea of how dissolution for classical logic works from Figure 3. The graph on the right is produced by dissolving on the link $\{A, \overline{A}\}$ in the graph on the left.

The two graphs are equivalent, but the one on the right contains three, not four, c-paths — the one path through the link is no longer present.

Path dissolution is in general applicable to collections of links. Here we restrict attention to single links, in which complementary literals A and $\overline{A}$ reside in conjoined subgraphs X and Y, respectively. With respect to the link $\{A,\ \overline{A}\}$ in Figure 3, $X = \begin{array}{c}\overline{C}\\ \wedge\\ D\end{array} \vee D$ and $Y = \overline{A} \vee \begin{array}{c}B\\ \wedge\\ C\end{array}$.

The *c-path complement* of A with respect to X, written $CC(A,\ X)$, is the subgraph of X consisting of all literals in X that lie on c-paths that do not contain A; the *c-path extension* of A with respect to X, written $CPE(A,\ X)$, is the subgraph containing all literals in X that lie on c-paths that *do* contain A. The c-paths through $(X \wedge Y)$ that do not contain the link are those through $(X \wedge CC(\overline{A},\ Y))$ (these are the paths of $(X \wedge Y)$ that do or do not include A and that *do not* include $\overline{A}$) plus those through $(CC(A,\ X) \wedge CPE(\overline{A},\ Y))$ (these are the paths that *do* include $\overline{A}$ but do not include A). The *dissolvent* of the link $\{A,\ \overline{A}\}$ in the subgraph M, denoted $DV(\{A,\ \overline{A}\},\ M)$, is defined to be

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

At every step, equivalence (and thus soundness) is preserved. Furthermore, the number of c-paths decreases by precisely the number of paths through the selected link. Thus termination is inevitable regardless of the choice of link at each

step. For unsatisfiable formulas, a graph without c-paths, the empty graph, must result. Satisfiable formulas always contain at least one c-path without a link, and such c-paths are preserved by dissolution; thus a non-empty graph without c-links will result, to which dissolution is not applicable. Any c-path in such a graph determines a set of models for the formula. It is this unavoidable termination that we call strong completeness. The reader is referred to [9] for more detail and for the formal definitions.

2.4 Signed path dissolution

Signed dissolution results from generalizing the standard formulation to operate on partial c-links. Suppose literals $S_1 : A$ and $S_2 : A$ reside in conjoined subgraphs X and Y. The standard dissolvent contains all paths of $X \wedge Y$ not containing $\{S_1 : A,\ S_2 : A\}$. We define the *signed path dissolvent* to be the standard dissolvent disjoined with that subgraph whose paths are exactly those that contain both $S_1 : A$ and $S_2 : A$. However, these latter paths can be simplified: The two literals in the partial link can be replaced by the single literal $(S_1 \cap S_2) : A$, by the Reduction Lemma.

Formally, let $M = X \wedge Y$, where $S_1 : A$ is in X and $S_2 : A$ is in Y. Then the *signed path dissolvent* of $L = \{S_1 : A,\ S_2 : A\}$ in M is $DV(L,\ M) =$

$$\begin{array}{ccc} X & & CC(S_1 : A,\ X) \\ \wedge & & \wedge \\ CC(S_2 : A,\ Y) & & CPE(S_2 : A,\ Y) \end{array} \quad \vee \quad \begin{array}{c} \{CPE(S_1 : A,\ X)\} - \{S_1 : A\} \\ \wedge \\ \{CPE(S_2 : A,\ Y)\} - \{S_2 : A\} \\ \wedge \\ (S_1 \cap S_2) : A \end{array} \quad \vee$$

The number of c-paths does not decrease unless $S_1 \cap S_2 = \emptyset$, in which case the rightmost disjunct in the dissolvent is dropped. Strong completeness, however, still holds; a finite number of steps will yield a graph without partial links. To make this paper more self-contained, we include the strong completeness result from [11].

Consider the set of *path-specific partial links* (PSPL's):

$$\{\ \langle l_p,\ p\rangle \mid l_p \text{is a partial link on } c\text{-path } p \text{ in } G\ \}\ .$$

All PSPL's in G corresponding to paths unchanged by a dissolution step are retained in the dissolvent. But paths in $CPE(S_1 : A,\ X) \wedge CPE(S_2 : A,\ Y)$ are replaced by paths that are identical except that they contain one less occurrence of atom A. As a result, the PSPL's corresponding to these paths have decreased in number. Therefore, finitely many dissolutions (bounded above by the number of PSPL's in the original graph) will yield a linkless equivalent graph. Notice that this analysis does not rely on the choice of partial link at each step.

We illustrate signed path dissolution with an example. Consider the Λ-atomic formula in Figure 4, where $\Delta = \{0,\ 1,\ 2\}$. We dissolve on $\{\{1,2\} : A,\ \{0,1\} : A\}$.

The dissolvent is shown in Figure 5.

If we continue dissolving until no partial links remain, a full dissolvent results (Figure 6). (Some of the dissolution steps are merely applications of the Reduction

$$\begin{array}{ccc} & & \{1\}:B \\ \{0,1\}:C & \vee & \wedge \\ & & \{1,2\}:A \\ & \wedge & \\ & & \{1\}:C \\ \{0,1\}:A & \vee & \wedge \\ & & \{0\}:B \end{array}$$

Figure 4:

$$\begin{array}{ccccccccc} & & \{1\}:B & & & & & & \\ \{0,1\}:C & \vee & \wedge & & & & & & \\ & & \{1,2\}:A & & & & \{0,1\}:C & & \{1\}:B \\ & \wedge & & & \vee & & \wedge & \vee & \wedge \\ & \{1\}:C & & & & & \{0,1\}:A & & \{1\}:A \\ & \wedge & & & & & & & \\ & \{0\}:B & & & & & & & \end{array}$$

Figure 5:

$$\begin{array}{ccccccc} \{1\}:C & & \{0,1\}:C & & \{1\}:B \\ \wedge & \vee & \wedge & \vee & \wedge \\ \{0\}:B & & \{0,1\}:A & & \{1\}:A \end{array}$$

Figure 6: A full signed dissolvent

Lemma — indeed, the one dissolution step applicable to $(S_1 : A \wedge S_2 : A)$ produces $S_1 \cap S_2 : A$ exactly.)

The preceding development of dissolution has been conducted from a refutation viewpoint. Just as unsatisfiability can be dualized to validity, so can these operators, and therefore dissolution itself, be dualized so as to focus on d-links and d-paths rather than on c-links and c-paths. Alternatively, such a disjunctive full dissolvent of G may be obtained using the version of dissolution defined above, operating on c-paths and partial c-links, simply by computing $\neg FD(\neg G)$. When there is a possibility of confusion, we will mention explicitly the type of link (and path) on which we are dissolving.

3 Prime Implicants/Implicates and Signed Formulas

3.1 Definitions

In this section we give basic definitions of prime implicants/implicates with respect to signed formulas. Note that a prime implicant of a classical formula is a weakest (or minimal) conjunction of literals implying the formula. Similarly, a prime s-implicant of a signed formula is a minimal conjunction of signed literals implying the signed formula, with respect to λ-consistency. This analogy can be dualized to one involving implicates and minimal disjunctions in the obvious way.

The signed formula G mentioned in these definitions need not be Λ-atomic, although typically it is.

A *conjunctive (disjunctive) term* is a conjunction (disjunction) of signed literals $S_i : x_i$ where each x_i appears exactly once.

A conjunctive term C *subsumes* another C' iff $C' \models_\Lambda C$.

A disjunctive term D *subsumes* another D' iff $D \models_\Lambda D'$.

Let C and C' be two conjunctive terms different from $\emptyset$.

A conjunctive term C is a *s-implicant* of a signed formula G, iff $C \models_\Lambda G$.

A *prime s-implicant* is an s-implicant not subsumed by another s-implicant.

Let D and D' be two disjunctive terms different from $\emptyset$.

A disjunctive term D is a *s-implicate* of a signed formula G, iff $G \models_\Lambda D$.

A *prime s-implicant* is an s-implicant not subsumed by another s-implicant.

Consider a c-path consisting of the signed literals $S_i : x_i$, $1 \leq i \leq n$. Essentially, it denotes the conjunction of its literal occurrences. Such a conjunction is more succinctly represented as follows: If there are distinct i and j such that $x_i = x_j$, then replace $S_i : x_i$ and $S_j : x_j$ by $(S_i \cap S_j) : x_i$ (by the Reduction Lemma).

$$\begin{array}{ccccccc} \{0,1\}:A & & \{0,2\}:A & & & & \\ \wedge & \vee & \wedge & \vee & \{1,2\}:A & \vee & \{0\}:B \\ \{2\}:B & & \{2\}:C & & & & \end{array}$$

Figure 7:

Iteration of this process will obviously produce a conjunctive term equivalent to the original c-path. If any of the resulting signs are empty, then the conjunctive term reduces to *false*; if any are equal to Δ, then that literal is dropped from the conjunctive term.

Similarly we can associate with any d-path an equivalent disjunctive term where signs of similar literals are combined by union rather than by intersection. If any of the resulting signs are equal to Δ, then the conjunctive term reduces to *true*; if any are empty, then that literal is dropped from the disjunctive term.

We say that a c-path (or a d-path) P *subsumes* another c-path (or d-path) P' iff the conjunctive term (or disjunctive term) equivalent to P subsumes the corresponding term for P'. Similarly, we may extend this notion to subsumption between c-paths (d-paths) and conjunctive (disjunctive) terms, or vice versa; in effect, whenever paths are present in a subsumption relationship, we coerce them to their corresponding terms, and apply the test to the terms.

The test for subsumption between conjunctive and disjunctive terms is a straightforward, albeit $\mathcal{O}(n^2)$ operation: A conjunctive term $C(\neq false)$ is *subsumed* by another conjunctive term C' if for every $S' : x \in C'$, there is a literal $S : x \in C$ such that $S \subseteq S'$.

A disjunctive term $C(\neq true)$ is *subsumed* by another disjunctive term C' if for every $S' : x \in C'$, there is a literal $S : x \in C$ such that $S \supseteq S'$.

As an example consider the signed formula in Figure 7, where $\Delta = \{0,1,2\}$.

An example of an s-implicant that is not a prime s-implicant is $\{0,2\} : A \wedge \{2\} : C$. However, $\{2\} : C$ is a prime s-implicant. Similarly $\{1,2\} : A \vee \{0,2\} : B \vee \{1,2\} : C$ is an s-implicate which is not prime, whereas $\{1,2\} : A \vee \{0,2\} : B \vee \{2\} : C$ is a prime s-implicate.

4 Prime S-Implicants

We now have the tools necessary for computing prime s-implicants (or s-implicates) of any Λ-atomic formula. (Assuming a finite Δ, of course.) As a result, these techniques can handle any finitely valued MVL in which formulas can be converted to Λ-atomic form. To find all prime s-implicants that force the condition that $\mathcal{F}$ evaluates to an element of S, find a Λ-atomic equivalent $\mathcal{F}'$ of $S : \mathcal{F}$, then find a (disjunctive) linkless equivalent $\mathcal{F}''$ of $\mathcal{F}'$, and finally collect the conjunctive terms of the c-paths of $\mathcal{F}''$ that are not subsumed by others.

In the subsection below, we establish the results that form the basis for this approach. In the following subsection, methods for obtaining s-implicants are defined precisely.

4.1 Foundations

Lemma 3 Every satisfiable c-path in a Λ-atomic formula G corresponds to an s-implicant of G.

Proof: Given a satisfiable c-path p in G, let C be the conjunctive term corresponding to p, and let I_s be a Λ-consistent interpretation which satisfies C. For each literal $S : x$ in C, there are literals $S_1 : x$, $S_2 : x$, ..., $S_k : x$ in p such that $S \subseteq S_i$, $1 \leq i \leq k$. Furthermore, all literals in p are so related to some literal in C. As a result, I_s satisfies every signed literal in p and hence satisfies G. □

Lemma 4 Let G be a semantic graph, and let H be a subgraph of G such that all c-paths through H are only partial c-paths in G. Then there exists a d-path q through G that does not meet H.

Proof: See [14]. (This lemma expresses a structural property of semantic graphs and was proved in [14]. The fact that signed formulas rather than standard propositional formulas are being represented has no effect on the lemma or on its proof.)□

Theorem 1 For any non-empty Λ-atomic formula G in which no d-path contains a partial link, every s-implicant of G is subsumed by some c-path of G.

Proof: Let $M = S_1 : x_1 \wedge \ldots \wedge S_n : x_n$ be an s-implicant of G. Since M $\models_\Lambda G$, $G \vee \neg M$ is valid with respect to Λ-consistency and is therefore spanned by its full set of (disjunctive) links, i.e., all d-paths contain a link. Moreover, only binary links need be considered because M is a conjunctive (and hence $\neg M$ is a disjunctive) term, and G is assumed to be free of partial links. Note that $\neg M = \overline{S_1} : x_1 \vee \ldots \vee \overline{S_n} : x_n$. Therefore, all d-links are of the form $\{\overline{S_i} : x_i, S_i' : x_i\}$, where $\overline{S_i} : x_i \in \neg M$, $S_i' : x_i$ is in G, and $\overline{S_i} \cup S_i' = \Delta$.

Let R be the set of all literals in G that are linked to $\neg M$. ¿From the observations above, we know that corresponding to each literal $S_i' : x_i$ in R, there is a literal $S_i : x_i \in M$ such that $S_i \subseteq S_i'$. So any c-path through R will subsume M, and, to prove the theorem, it suffices to show that G_R contains a c-path through G.

So suppose that every c-path in G_R is partial in G. By Lemma 4 there is a d-path p through G which does not meet G_R. The path p is not linked to $\neg M$ because it has no nodes from R; furthermore, p itself contains no links since G is linkless. Therefore, the d-path comprised of the nodes from both p and $\neg M$ is a linkless d-path through $G \vee \neg M$. But this is a contradiction: $G \vee \neg M$ is spanned by its d-links by definition. As a result, some c-path in G_R is not partial in G and the proof is complete. □

Theorem 2 Suppose G is a non-null graph representing a Λ-atomic formula having no partial d-links, and let $\psi(G)$ be defined as follows:

$$\psi(G) = \{p \mid (p \; is \; a \; c{-}path \; through \; G) \wedge (p \neq false) \wedge (\forall \; c{-}paths \; q \; through \; G, \; q \; does \; not \; subsume \; p)\} .$$

Then ψ(G) is the set of all prime s-implicants of G.

Proof: Direct result of Lemma 3 and Theorem 1. □

We say that a signed formula is in *reduced CNF* if each of its clauses are disjunctive terms, i.e., if it has no *true* disjuncts, and every disjunct has at most one signed literal for each variable.

Corollary 1 *If* G *is a signed formula in reduced CNF, then* $\psi(G)$ *is the set of all prime s-implicants of* G.

This follows from Theorem 1 as such a CNF formula has no d-paths with partial links. We mention this corollary for historical reasons: The classical version is simply Nelson's theorem [12]. Of course, it is of limited computational interest since moving from NNF to CNF may entail an exponential penalty.

Theorem 3 $FD(\mathcal{F})$ (with respect to c-links or to d-links) is equivalent to $\mathcal{F}$ under all Λ-consistent interpretations.

Proof: See [11]. □

The removal of all partial d-links prior to collecting the c-paths is crucial to our method; in particular, Theorem 1 cannot be strengthened into the converse of Lemma 3. The formula (where $\Delta = \{0, 1, 2\}$)

$$\begin{array}{ccc} \{1\} : C & & \{0,2\} : C \\ \wedge & \vee & \wedge \\ \{0\} : B & & \{1,2\} : A \end{array}$$

has one d-link and has $\{\{0\} : B, \{1,2\} : A\}$ as a prime-s implicant, which is not contained in any c-path.

4.2 Computing s-implicants

We can now state the steps in finding all the prime s-implicants of a signed formula $\mathcal{F}$.

Step 1: If $\mathcal{F}$ is not Λ-atomic, push the sign inward resulting in an equivalent Λ-atomic formula $\mathcal{F}'$ (Section 5).

Step 2: Find $\mathcal{F}''$, a Λ-atomic formula equivalent to $\mathcal{F}'$ that has no d-links, by computing $FD(\mathcal{F}')$ with respect to d-links or by converting to reduced CNF.

Step 3: Find all the c-paths of $\mathcal{F}''$ that are not subsumed by other c-paths. The conjunctive terms corresponding to these c-paths are the prime s-implicants of $\mathcal{F}$.

The correctness of the steps follows from Theorems 2 and 3.

We illustrate the different steps by an example. Consider the semantic graph from Figure 7:

$$\begin{array}{ccccccccc} \{0,1\} : A & & \{0,2\} : A & & & & & & \\ \wedge & \vee & \wedge & \vee & \{1,2\} : A & \vee & \{0\} : B & & \\ \{2\} : B & & \{2\} : C & & & & & & \end{array}$$

This is in Λ-atomic form so we do nothing in Step 1. In Step 2 we find the full dissolvent with respect to d-links. Dissolving on the $\{\{0,1\} : A, \{1,2\} : A\}$ link amounts to replacing $\{0,1\} : A$ by $\Delta : A = \mathit{true}$; similarly for the $\{\{0,2\} : A, \{1,2\} : A\}$ link. This leaves one d-path containing the $\{\{2\} : B, \{0\} : B\}$ link. This is a very special case where dissolution reduces to a simple application of the Reduction Lemma. The resulting full (disjunctive) dissolvent is:

$$\{1,2\} : A \quad \vee \quad \{0,2\} : B \quad \vee \quad \{2\} : C \quad .$$

Since none of the c-paths subsume each each other, the prime s-implicants are exactly the c-paths of the graph: $\{1,2\} : A$, $\{0,2\} : B$ and $\{2\} : C$.

We have so far restricted our discussion to finding prime s-*implicants* only. However we can dualize all the theorems and lemmas. This would give us a method to find prime s-*implicates*. Alternatively, given G, one could find the prime s-implicants of $\neg G$ and then negate them to produce the prime s-implicates of G.

5 Post Logics

In this section we show how one can find all the prime implicants of arbitrary functions of a Post logic. The key observation is that this can be accomplished using signed formulas in which all signs are *regular* (defined below). For any signed formula $S : \mathcal{F}$, where S is regular and the logic Λ of $\mathcal{F}$ is a Post logic (and hence expressible as a regular logic), we have reasonable techniques for computing an equivalent Λ-atomic formula.

5.1 Regularity

We assume that Δ is a finite linearly ordered set $\{0, 1, 2, \ldots, n-1\}$. We say that S is a *regular sign* if it represents an interval of Δ containing either 0 or $n-1$. Obviously, any regular sign can be represented as $\{< i\}$ or $\{> i\}$, where $i \in \Delta$ (or as Δ or $\emptyset$ in case they are needed). When a regular sign is pushed inside a regular operator, the arguments of the operator becomes signed, and those resulting signs are also regular.

The connectives include $\wedge_p = \mathbf{min}$ and $\vee_p = \mathbf{max}$. We also have the unary operator σ defined by $\sigma(i) = i + 1 \, mod \, n$. Recall that a regular unary operator μ is one that is monotonic. Although σ is not monotonic and hence not regular, it can be expressed using regular operators: $\sigma(i) = \sigma'(i) \wedge_p \overline{J_{n-1}}(i)$, where $\sigma'(i) = \mathbf{min}(i+1, n-1)$ and $\overline{J_{n-1}}(i) = n - 1 - i + (i \, mod \, n - 1)$.

Given a signed formula $S : \mathcal{F}$, where sign S is regular and $\mathcal{F}$ is a formula from a Post logic, we have the following rules for driving S inward.

$$
\begin{array}{rcl}
\{< i\} : (\mathcal{G} \wedge_p \mathcal{H}) & \equiv_\Lambda & \{< i\} : \mathcal{G} \vee \{< i\} : \mathcal{H} \\
\{> i\} : (\mathcal{G} \wedge_p \mathcal{H}) & \equiv_\Lambda & \{> i\} : \mathcal{G} \wedge \{> i\} : \mathcal{H} \\
\{< i\} : (\mathcal{G} \vee_p \mathcal{H}) & \equiv_\Lambda & \{< i\} : \mathcal{G} \wedge \{< i\} : \mathcal{H} \\
\{> i\} : (\mathcal{G} \vee_p \mathcal{H}) & \equiv_\Lambda & \{> i\} : \mathcal{G} \vee \{> i\} : \mathcal{H} \\
\{> i\} : \sigma'(\mathcal{F}) & \equiv_\Lambda & \{> i-1\} : \mathcal{F} \\
\{< i\} : \sigma'(\mathcal{F}) & \equiv_\Lambda & \{< i-1\} : \mathcal{F} \\
\{> i\} : \overline{J_{n-1}}(\mathcal{F}) & \equiv_\Lambda & \{< n-1\} : \mathcal{F} \\
\{< i\} : \overline{J_{n-1}}(\mathcal{F}) & \equiv_\Lambda & \{> n-2\} : \mathcal{F}
\end{array}
$$

In applying the above rules, we assume the following simplifications:
$\{< 0\} : \mathcal{F}$ and $\{> n-1\} : \mathcal{F}$ are always *false*;
$\{< n\} : \mathcal{F}$ and $\{> -1\} : \mathcal{F}$ are always *true*.

The notational conventions for expressing prime implicants and prime implicates of formulas in Post logics differs from our general notation for MVL's. In particular, these notions are approached in the context of the MVL itself, whereas our approach is through the use of the meta-language of signed formulas. In the remainder of this section, we relate concepts and notation from Post logics to signed formulas. Many of the following definitions and notation are taken from [2].

5.2 Product terms and s-implicants

An interpretation I maps variable x_i to an element of $\Delta = \{0, 1, \ldots, n-1\}$. A *literal* is of the form $x_i(a, b)$, where x_i is a variable and $0 \leq a, b \leq n-1$. Interpretation I maps literal $x_i(a, b)$ to $n-1$ if $a \leq I(x_i) \leq b$, 0 otherwise. A *product term* is of the form $K \wedge_p x_1(a_1, b_1) \wedge_p \ldots \wedge_p x_m(a_m, b_m)$, where $1 \leq K \leq n-1$, and $x_i(a_i, b_i)$ is a literal. Such a product term will always evaluate either to K or to 0. If a product term does not include variable x_j, then we can find an equivalent product term which includes x_j by adding the literal $x_j(0, n-1)$. In the rest of this paper we assume that product terms include all variables under consideration.

A product term R is an implicant of a Post logic truth function F if for every interpretation I, $I(R) \leq I(F)$. Let R be the product term $K \wedge_p x_1(a_1, b_1) \wedge_p \ldots \wedge_p x_m(a_m, b_m)$, and let R' be the term $K' \wedge_p x_1(c_1, d_1) \wedge_p \ldots \wedge_p x_m(c_m, d_m)$. We say that R subsumes R' if $K' \leq K$ and, for each i, $a_i \leq c_i \leq d_i \leq b_i$. Note that the intervals associated with the variables of product terms do not necessarily correspond to regular signs. This is to be expected: Although we can preserve regularity while generating a Λ-atomic formula, processing the formula with either signed dissolution or conversion to reduced CNF will produce non-regular signs. However, each such sign is a finite set, and can therefore be be represented as a unique minimal set of disjoint intervals. For example, the sign $\{0, 1, 3, 4, 6\}$ can be viewed as the set of intervals $\{(0, 1), (3, 4), (6, 6)\}$; we say that (0,1), (3,4), and (6,6) are the intervals *corresponding* to the sign $\{0, 1, 3, 4, 6\}$.

Let $\{m | m \; is \; a \; prime \; s{-}implicant \; of \; > i : F\}$ be denoted by $\mathcal{S}_{\{>i\}:F}$. Let $M = S_1 : x_1 \wedge \ldots \wedge S_m : x_m$ be an s-implicant of $\{> i\} : F$. We define $\mathcal{R}_M$ to be the set of all product terms of the form $(i+1) \wedge x_1(a_1, b_1) \ldots \wedge x_m(a_m, b_m)$, where for $1 \leq i \leq m$, (a_i, b_i) is an interval corresponding to S_i. (If x_i does not

appear in M, then $a_i = 0$ and $b_i = n-1$.) For example, let $\{1,2,4\} : x_1 \wedge \{3\} : x_2$ be an s-implicant of $\{> 3\} : F$. Then the corresponding product terms are $4 \wedge_p x_1(1,2) \wedge x_2(3,3)$ and $4 \wedge_p x_1(4,4) \wedge x_2(3,3)$.

It is easy to see that none of the product terms in $\mathcal{R}_M$ subsume each other.

Theorem 4 Let F be a formula, $\mathcal{S}_{\{>i\}:F}$ be the set of all prime s-implicants of $\{> i\} : F$, and $\mathcal{R}_{\{>i\}:F}$ be the union of all the sets $\mathcal{R}_{M_i}$, where $M_i \in \mathcal{S}_{\{>i\}} : F$. Then every product term in $\mathcal{R}_{\{>i\}:F}$ is an implicant of F.

Proof: Let R be a product term in $\mathcal{R}_{\{>i\}:F}$, and let M be the corresponding s-implicant. Every interpretation over Λ maps R either to 0 or to $i+1$. If $I(R) = 0$, then obviously $I(R) \leq I(F)$.

So suppose $I(R) = i+1$. Then I must map every literal $x_j(a_j, b_j)$ in the product term to n-1; we have $a_j \leq I(x_j) \leq b_j$. Let I_s be the corresponding Λ-consistent interpretation; by definition $I_s(S_j : x_j) = \mathit{true}$ and $\{a_j, a_j + 1, \ldots, b_j\} \subseteq S_j$. Hence $I_s(\{> i\} : F)$ is also *true*, $I(F) \geq i + 1$, and R is an implicant of F. □

Theorem 5 Suppose F is a formula and product term R is an implicant of F, where K is the constant in R. Then there is some product term in $\mathcal{R}_{\{>K-1\}:F}$ which subsumes R.

Proof: Let $R = K \wedge_p x_1(a_1, b_1) \wedge_p \ldots \wedge_p x_m(a_m, b_m)$, and consider $M = S_1 : x_1 \wedge \ldots \wedge S_m : x_m$, where S_j is the set of elements in the range a_j to b_j. M is an s-implicant of $\{> K-1\} : F$ and will be subsumed by some prime s-implicant $S'_1 : x_1 \wedge \ldots \wedge S'_m : x_m$ in $\mathcal{S}_{\{>K-1\}:F}$ (say M'), where $S'_i \supseteq S_i$, $1 \leq i \leq m$. Now consider the product R' in $\mathcal{R}_{M'}$ that is of the form $K \wedge_p x_1(a'_1, b'_1) \wedge_p \ldots \wedge_p x_m(a'_m, b'_m)$ such that $a'_j \leq a_j \leq b_j \leq b'_j$ for $1 \leq j \leq m$. There must be such a product term since $S'_j \supseteq S_j$ for each j and hence R' subsumes R. □

Theorem 6 Let F be a formula, and let $\mathcal{K} = \bigcup_{i=0}^{n-2} \mathcal{R}_{\{>i\}:F}$. Let $\mathcal{I}$ be the result of removing all product terms in $\mathcal{K}$ that are subsumed by others. Then $\mathcal{I}$ is the set of all prime implicants of F.

Proof: Follows directly from Theorems 4 and 5. □

We can now give a method to find the set of prime implicants for a given formula F.

Step 1: Set $\mathcal{I}$ to $\emptyset$.

Step 2: For each i, from $n-2$ down to 0, do

- **Step 2a:** Find $\mathcal{S}$, the set of prime s-implicants of $\{> i\} : F$.
- **Step 2b:** Find $\mathcal{R}_{\mathcal{S}}$, the set of all product terms corresponding to prime s-implicants of $\{> i\} : F$.
- **Step 2c:** Add to $\mathcal{I}$ all those product terms in $\mathcal{R}_{\mathcal{S}}$ not subsumed by other product terms in $\mathcal{I}$.

At the end of Step 2, $\mathcal{I}$ contains exactly the prime implicants of F. The correctness of the steps follows from Theorem 6.

5.3 An example

Let $\Delta = \{0,1,2\}$ and consider the following formula: $\mathcal{F} = (\sigma(x_1) \wedge_p \sigma(x_3)) \vee_p (x_2 \wedge_p x_3)$ Since $n = 3$, Step 2 above need only be iterated for $i = 1, 0$; we show the case $i = 1$ in detail.
To compute the prime s-implicants of $\{>1\} : \mathcal{F}$, first push the sign inward:

$$\{>1\} : \mathcal{F} = \{>1\} : (\sigma(x_1) \wedge_p \sigma(x_3)) \vee_p (x_2 \wedge_p x_3)$$

$$\begin{aligned}
&= \{>1\} : (\sigma(x_1) \wedge_p \sigma(x_3)) \vee \{>1\} : (x_2 \wedge_p x_3) \\
&= (\{>1\} : \sigma(x_1) \wedge \{>1\} : \sigma(x_3)) \vee (\{>1\} : x_2 \wedge \{>1\} : x_3) \\
&= (\{>1\} : (\sigma'(x_1) \wedge_p \overline{J_2}(x_1)) \\
&\qquad \wedge \{>1\} : (\sigma'(x_3) \wedge_p \overline{J_2}(x_3))) \vee (\{2\} : x_2 \wedge \{2\} : x_3) \\
&= (\{>1\} : (\sigma'(x_1) \wedge \{>1\} : \overline{J_2}(x_1)) \wedge \{>1\} : (\sigma'(x_3) \\
&\qquad \wedge \{>1\} : \overline{J_2}(x_3))) \vee (\{2\} : x_2 \wedge \{2\} : x_3) \\
&= ((\{>0\} : x_1 \wedge \{<2\} : x_1) \\
&\qquad \wedge (\{>0\} : x_3 \wedge \{<2\} : x_3)) \vee (\{2\} : x_2 \wedge \{2\} : x_3) \\
&= (\{1\} : x_1 \wedge \{1\} : x_3) \vee (\{2\} : x_2 \wedge \{2\} : x_3)
\end{aligned}$$

Note that the last step above is due to the Reduction Lemma, not to the rules for distributing signs over connectives; the latter preserve regularity. As a result, the sign $\{1\}$ (for both x_1 and x_3) is not regular.

Below we write in two-dimensional format, the Λ-atomic formula above:

$$\begin{array}{ccc}
\{1\} : x_1 & & \{2\} : x_2 \\
\wedge & \vee & \wedge \\
\{1\} : x_3 & & \{2\} : x_3
\end{array}$$

The only partial d-link is $\{\{1\} : x_3, \{2\} : x_3\}$; dissolving (disjunctively) on this link produces a full dissolvent.

$$\begin{array}{ccc}
\{1\} : x_1 & & \\
\wedge & \vee & \{2\} : x_2 \\
\{1\} : x_3 & & \\
& \wedge & \\
\{1\} : x_1 & \vee & \{2\} : x_3 \\
& \wedge & \\
& \{1,2\} : x_3 &
\end{array}$$

Note that the c-path containing both $\{1\} : x_3$ and $\{2\} : x_3$ reduces to $false$ by the Reduction Lemma. The other three c-paths reduce to the following s-implicants.

$$\begin{array}{ccc}
\{1\} : x_1 & \{2\} : x_2 & \{2\} : x_2 \\
\wedge & \wedge & \wedge \\
\{1\} : x_3 & \{1\} : x_1 & \{2\} : x_3 \\
& \wedge & \\
& \{1,2\} : x_3 &
\end{array}$$

None of these subsume each other and are thus all prime s-implicants. None of the corresponding product terms subsume each other, and all are added to $\mathcal{I}$.

$$\begin{array}{ccc} 2 & 2 & 2 \\ \wedge_p & \wedge_p & \wedge_p \\ x_1(1,1) & x_2(2,2) & x_2(2,2) \\ \wedge_p & \wedge_p & \wedge_p \\ x_3(1,1) & x_1(1,1) & x_3(2,2) \\ & \wedge_p & \\ & x_3(1,2) & \end{array}$$

Now Step 2 must be repeated with $i = 0$; it is left to the reader to verify that this produces the following Λ-atomic formula.

$$\begin{array}{ccc} \{0,1\} : x_1 & & \{1,2\} : x_2 \\ \wedge & \vee & \wedge \\ \{0,1\} : x_3 & & \{1,2\} : x_3 \end{array}$$

Dissolving on the partial d-link $\{\{0,1\} : x_3, \{1,2\} : x_3\}$ produces a full dissolvent for $\{> 0\} : \mathcal{F}$.

$$\begin{array}{ccc} \{0,1\} : x_1 & & \\ \wedge & \vee & \{1,2\} : x_2 \\ \{0,1\} : x_3 & & \\ & \wedge & \\ \{0,1\} : x_1 & \vee & \{1,2\} : x_3 \end{array}$$

In this case, none of the four c-paths is contradictory; the following s-implicants are obtained via the Reduction Lemma.

$$\begin{array}{cccc} \{0,1\} : x_1 & \{0,1\} : x_1 & \{1,2\} : x_2 & \{1,2\} : x_2 \\ \wedge & \wedge & \wedge & \wedge \\ \{0,1\} : x_3 & \{1\} : x_3 & \{0,1\} : x_1 & \{1,2\} : x_3 \end{array}$$

We get the prime s-implicants by removing the second s-implicant above, which is subsumed by the first. The corresponding product terms must also be checked for subsumption by those terms already in $\mathcal{I}$. In this particular example, none are. The values of $\mathcal{S}$,$\mathcal{R}_\mathcal{S}$,$\mathcal{I}$ at the end of each iteration of Step 2 are shown in the table below.

$$
\begin{array}{ccc}
\{> i-1\} : x_1 & & \\
\wedge & & \\
\{< n-1\} : x_1 & & \{> i\} : x_2 \\
\wedge & \vee & \wedge \\
\{> i-1\} : x_3 & & \{> i\} : x_3 \\
\wedge & & \\
\{< n-1\} : x_3 & &
\end{array}
$$

Figure 8: Driving parameterized signs inward

i	$\mathcal{S}$	$\mathcal{R}_{\mathcal{S}}$	$\mathcal{I}$
1	$\{1\} : x_1 \wedge \{1\} : x_3$	$2 \wedge_p x_1(1,1) \wedge_p x_3(1,1)$	$2 \wedge_p x_1(1,1) \wedge_p x_3(1,1)$
	$\{2\} : x_2 \wedge \{1\} : x_1$	$2 \wedge_p x_2(2,2) \wedge_p x_1(1,1)$	$2 \wedge_p x_2(2,2) \wedge_p x_1(1,1)$
	$\wedge \{1,2\} : x_3$	$\wedge_p x_3(1,2)$	$\wedge_p x_3(1,2)$
	$\{2\} : x_2 \wedge \{2\} : x_3$	$2 \wedge_p x_2(2,2) \wedge_p x_3(2,2)$	$2 \wedge_p x_2(2,2) \wedge_p x_3(2,2)$
0	$\{0,1\} : x_1 \wedge \{0,1\} : x_3$	$1 \wedge_p x_1(0,1) \wedge_p x_3(0,1)$	$2 \wedge_p x_1(1,1) \wedge_p x_3(1,1)$
	$\{1,2\} : x_2 \wedge \{0,1\} : x_1$	$1 \wedge_p x_2(1,2) \wedge_p x_1(0,1)$	$2 \wedge_p x_2(2,2) \wedge_p x_1(1,1)$
			$\wedge_p x_3(1,2)$
	$\{1,2\} : x_2 \wedge \{1,2\} : x_3$	$1 \wedge_p x_2(1,2) \wedge_p x_3(1,2)$	$2 \wedge_p x_2(2,2) \wedge_p x_3(2,2)$
			$1 \wedge_p x_1(0,1) \wedge_p x_3(0,1)$
			$1 \wedge_p x_2(1,2) \wedge_p x_1(0,1)$
			$1 \wedge_p x_2(1,2) \wedge_p x_3(1,2)$

The process terminates after two iterations and $\mathcal{I}$ has all the prime implicants of F.

5.4 Improvements

There are several aspects of this approach that may be significantly improved. Consider Step 2a in the algorithm as defined above. By recomputing these steps for each value of i, the sign $\{> i\}$ is pushed inward at each iteration. However, it is easy to see that the variable i may be left in the sign, and a Λ-atomic equivalent can be produced that is essentially parameterized in terms of i. (The careful reader will have noted that the rules for pushing signs inward in Section 5.1 were indeed given in this form.) A similar "parameterization" of signs has been employed in [4].

In the example above, this parameterization would result in the Λ-atomic formula of Figure 8.
Even in this case, the Reduction Lemma can be applied twice to the left conjunct. The resulting signs for both x_1 and x_3 contain the elements in the interval $[i..n-2]$. For simplicity, we denote this sign by $\{i..n-2\}$.

So suppose we have the conjunction $[\wedge_{i=1}^{m}\{> e_i\} : X] \wedge [\wedge_{i=m+1}^{n}\{< e_i\} : X]$. Let $e^c_{max} = \mathbf{max}_{i=1}^{m}(e_i + 1)$ and let $e^c_{min} = \mathbf{min}_{i=m+1}^{n}(e_i - 1)$. Then by the Reduction Lemma, we can express this conjunction as $\{e^c_{max}..e^c_{min}\} : X$. Note that if $m = 0$, $e^c_{max} = 0$; if $n = m$, $e^c_{min} = n - 1$. Also, if $e^c_{max} > e^c_{min}$, then $\{e^c_{max}..e^c_{min}\}$: $X = \emptyset : X = false$.

Now consider the disjunction $[\vee_{i=1}^{m}\{> e_i\} : X] \vee [\vee_{i=m+1}^{n}\{< e_i\} : X]$. Let $e^d_{max} = \mathbf{max}_{i=m+1}^{n}(e_j)$ and let $e^d_{min} = \mathbf{min}_{i=1}^{m}(e_j)$. This disjunction can be reduced to $\overline{\{e^d_{max}..e^d_{min}\}} : X$. Here, if $n = m$, $e^d_{max} = 0$; if $m = 0$, $e^d_{min} = n - 1$. Also, if $e^d_{max} > e^d_{min}$, then $\overline{\{e^d_{max}..e^d_{min}\}} : X = \overline{\emptyset} : X = \Delta : X = true$. Note also that $\overline{\{e^d_{max}..e^d_{min}\}} : X = (\{0..e^d_{max} - 1\} \cup \{e^d_{min} + 1..n - 1\}) : X$.

Thus, if we apply the Reduction Lemma to a conjunction in which the signs are regular, the resulting sign is a single interval in the range $0..n - 1$ that can be either empty or non-regular. Similarly, if we apply the Reduction Lemma to a disjunction in which the signs are regular, the resulting sign is the complement of a (possibly empty or non-regular) single interval in the range $0..n - 1$. It is easy to see, however, from the rules of Section 5.1, that the e_j's will all be linear expressions in terms of i and n. So the interval limits are reducible to simple expressions in terms of i and n, and we have proved:

Theorem 7 Let $\mathcal{F}$ be a formula in a Post logic, where $\Delta = \{0, 1, 2, \ldots, n-1\}$, and let $\mathcal{F}'$ be a Λ-atomic equivalent of $\{> i\} : \mathcal{F}$ produced by the rules for distributing regular signs over the connectives. Let $\mathcal{F}''$ be the result of applying the Reduction Lemma to $\mathcal{F}'$ wherever possible. Then $\mathcal{F}'$ is regular, and the signs of $\mathcal{F}''$ can be expressed as single intervals $\{e_1..e_2\}$ or their complements, where e_1, e_2 are linear expressions in terms of i and n. □

A more subtle improvement (but one that may have a more dramatic effect on efficiency) results from leaving the signs unspecified *during the dissolution process.* Thus we compute the full dissolvent only once rather than at each iteration. It turns out that signed dissolution manipulates signs in precisely the same manner as does the Reduction Lemma. Recall that given $M = X \wedge Y$, where $S_1 : A$ is in X and $S_2 : A$ is in Y, the *signed path dissolvent* of the partial link $\{S_1 : A, S_2 : A\}$ in M is

$$\begin{array}{ccccc} X & & CC(S_1 : A,\ X) & & \{CPE(S_1 : A,\ X)\} - \{S_1 : A\} \\ \wedge & \vee & \wedge & \vee & \wedge \\ CC(S_2 : A,\ Y) & & CPE(S_2 : A,\ Y) & & \{CPE(S_2 : A,\ Y)\} - \{S_2 : A\} \\ & & & & \wedge \\ & & & & (S_1 \cap S_2) : A \end{array} .$$

Notice that the only subformula of the signed dissolvent in which a new sign is introduced is $(S_1 \cap S_2) : A$. But this is simply $S_1 : A \wedge S_2 : A$, by the Reduction Lemma.

The signed dissolvent has been defined using the intersection of signs S_1 and S_2 assuming the signs to be constant expressions. Under this assumption, $S_1 \cap S_2$ can be computed as a single sign immediately. But here we wish to keep signs

parameterized in terms of i and n while driving them inward. To more conveniently represent this parameterized state during the dissolution process, we may express the signed dissolvent in a slightly different form:

$$\begin{array}{ccccc} X & & CC(S_1 : A,\, X) & & \{CPE(S_1 : A,\, X)\} - \{S_1 : A\} \\ \wedge & \vee & \wedge & \vee & \wedge \\ CC(S_2 : A,\, Y) & & CPE(S_2 : A,\, Y) & & \{CPE(S_2 : A,\, Y)\} - \{S_2 : A\} \\ & & & & \wedge \\ & & & & S_1 : A \\ & & & & \wedge \\ & & & & S_2 : A \end{array} .$$

This version of the signed dissolvent makes explicit the essence of the operation. The graph is rewritten in a manner that isolates the c-paths containing the partial link: $CPE(S_1 : A,\, X)$ is a conjunction in which $S_1 : A$ is a conjunct; similarly for $CPE(S_2 : A,\, Y)$. The rightmost disjunct in the dissolvent is a conjunction capturing the paths containing the partial link, in which the literals of the partial link have merely been brought together via associativity and commutativity of conjunction. Since we need not reassociate and commute until the Reduction Lemma is actually applied, a further reformulation of signed dissolution is appropriate.

$$\begin{array}{ccccc} X & & CC(S_1 : A,\, X) & & CPE(S_1 : A,\, X) \\ \wedge & \vee & \wedge & \vee & \wedge \\ CC(S_2 : A,\, Y) & & CPE(S_2 : A,\, Y) & & CPE(S_2 : A,\, Y) \end{array}$$

Here we have avoided any set operations on signs and simply restructured the graph so as to isolate paths containing the partial link. But this is exactly what is needed to make the Reduction Lemma applicable to the literals of the partial link. For the purposes of this paper, we call this formulation of signed dissolution *lazy signed dissolution*.

Observe that all of the above formulations of signed dissolution are equivalent. The only apparent distinction is in how the potential sign intersection operations are represented. In an implementation, it may be advantageous to employ such "lazy evaluation" only partially and to include some amount of look ahead to recognize when sign intersections have become empty, for example. There is an operational distinction in these formulations, however, that must be handled.

The strong termination properties of signed dissolution depend upon producing a graph without partial links. In particular, PSPL's are strictly reduced at each step. By not explicitly combining the two literals of a partial link, we leave the number of PSPL's unchanged, and this termination condition will not arise. However, the problem can be overcome with some simple bookkeeping.

Essentially, a partial link to which the Reduction Lemma is already applicable should be ignored. This is because lazy signed dissolution acts almost as the identity operator on such a link; its literals are members of a single conjunction which is essentially left unchanged. However, a partial link to which the Reduction Lemma is not applicable is transformed by lazy signed dissolution; the Reduction Lemma does become applicable to its literals in the dissolvent as a result. Thus termination will occur when the Reduction Lemma is applicable to all partial links.

Theorem 8 For propositional signed formulas, lazy signed dissolution with partial links is a strongly complete rule of inference, when restricted to links to which the Reduction Lemma does not apply.

Proof: We partition the set of PSPL's into two blocks: Let B_{RL} be the block containing those to which the Reduction Lemma is applicable, and let $B_{\overline{RL}}$ contain those to which it is not. By dissolving only on partial links whose corresponding PSPL's are from $B_{\overline{RL}}$, we add those PSPL's to B_{RL}, and remove them from $B_{\overline{RL}}$. Since total PSPL's is an invariant under lazy signed dissolution, $B_{\overline{RL}}$ will become empty after a finite number of steps. □

It should be noted that the development above regarding signed dissolution on partial c-links can be dualized for signed dissolution on partial d-links. In the latter case, evaluation of unions of signs is delayed, rather than intersections, for the computation of prime implicants, rather than implicates.

It is apparent from the discussion above that a full signed dissolvent can be computed in which all signs are present in the Λ-atomic formula with which we began; those signs may be regular signs expressed in terms of i and n, where $0 \leq i \leq n$. As a result, we have the following generalization of Theorem 8:

Theorem 9 Let $\mathcal{F}$ be a formula in a Post logic, where $\Delta = \{0, 1, 2, \ldots, n-1\}$. Let $\mathcal{F}'$ be a full dissolvent of a Λ-atomic equivalent of $\{> i\} : \mathcal{F}$, produced by first applying the rules for distributing regular signs over the connectives, and then by applying lazy signed dissolution. Let $\mathcal{F}''$ be the result of applying the Reduction Lemma to $\mathcal{F}'$ wherever possible. Then $\mathcal{F}'$ is regular, and the signs of $\mathcal{F}''$ can be expressed as single intervals $\{e_1..e_2\}$ or their complements, where e_1, e_2 are linear expressions in terms of i and n. □

We can now state an improved method to find the set of prime implicants for a given formula F.

Step 1: Set $\mathcal{I}$ to $\emptyset$.

Step 2: Compute F', a full dissolvent of a Λ-atomic equivalent of $\{> i\} : F$.

Step 3: Compute F'' by applying the Reduction Lemma to F' wherever possible.

Step 4: For each i, from $n-2$ down to 0, do

Step 4a: Find $\mathcal{S}$, the set of prime s-implicants of F''.

Step 4b: Find $\mathcal{R}_{\mathcal{S}}$, the set of all product terms corresponding to prime s-implicants of F''.

Step 4c: Add to $\mathcal{I}$ all product terms in $\mathcal{R}_{\mathcal{S}}$ not subsumed by other product terms in $\mathcal{I}$.

At the end of Step 4, $\mathcal{I}$ contains exactly the prime implicants of F. The correctness of the algorithm follows from Theorem 9.

Note that in the above algorithm, pushing signs inward and dissolving is expressed explicitly, outside the loop. In the method presented initially, these computations were implicit in Step 2a for finding the set of prime s-implicants, and were thus inside the loop. Observe also that the computation of prime s-implicants was described in Section 4.2 as a three step process. Here, only the third step is required in which we find all the c-paths of F'' that are not subsumed by other c-paths. The conjunctive terms corresponding to these c-paths are the prime s-implicants of $\{> i\} : F$.

Consider the parameterized Λ-atomic formula of Figure 8, further simplified via the Reduction Lemma.

$$\begin{array}{ccc} \{i..n-2\} : x_1 & & \{> i\} : x_2 \\ \wedge & \vee & \wedge \\ \{i..n-2\} : x_3 & & \{> i\} : x_3 \end{array}$$

Now the sole d-link $\{\{i..n-2\} : (x_3), \{> i\} : x_3\}$ is in parameterized form, and we compute the (disjunctive) lazy signed dissolvent.

$$\begin{array}{ccc} \{i..n-2\} : x_1 & & \\ \wedge & \vee & \{i+1..n-1\} : x_2 \\ \{i..n-2\} : x_3 & & \\ & \wedge & \\ \{i..n-2\} : x_1 & \vee & \{i+1..n-1\} : x_3 \\ & \wedge & \\ \{i..n-2\} : x_3 & \vee & \{i+1..n-1\} : x_3 \end{array}$$

Since the Reduction Lemma applies to the only d-link now present, lazy dissolution terminates, the lemma is applied, and of course we know that $n = 3$.

$$\begin{array}{ccc} \{i..1\} : x_1 & & \\ \wedge & \vee & \{i+1..2\} : x_2 \\ \{i..1\} : x_3 & & \\ & \wedge & \\ \{i..1\} : x_1 & \vee & \{i+1..2\} : x_3 \\ & \wedge & \\ & \{i..2\} : x_3 & \end{array}$$

This completes Step 3 of the improved algorithm, and only at this point do we initiate the loop for $i = 1$ down to 0. Merely instantiating i in the formula above produces the two formulas created from scratch at Step 2a of the original algorithm. The improved algorithm must, for each iteration, still compute the prime s-implicants, product terms, and prime implicants for each instance of i. This computation appears unavoidable and is identical in both versions.

6 Conclusions and Future Work

The current study has provided, in addition to earlier work on prime implicant methods for MVL's based on truth table analysis, a formula-based approach us-

ing the logic of signed formulas. In the attempt to improve Step 2a of the initial method, we have shown how signs may be pushed inward while left as expressions parameterized in terms of the elements and cardinality of Δ. A more subtle improvement has been introduced in which signs are left in parameterized form during the inference process. This has been carried out for signed dissolution in the presence of regular signs.

A good question for future research is whether these techniques can be extended in various ways. Most inference techniques are applicable to signed formulas in a relatively straightforward way. Thus extending these results to computations based on alternative inference rules (e.g., signed resolution) should also be straightforward. A more challenging goal would be to determine ways in which the regularity requirement on signs could be relaxed. Alternatively, by retaining regularity, perhaps the restriction of finiteness on Δ could be dropped.

References

[1] Allen, C.M. The Allen-Givone implementation oriented algebra. In D.C Rine editor, *Computer Science and multiple-valued logic*, North Holland 1984, 268-288.

[2] Hähnle, R. Uniform notation tableau rules for multiple-valued logics. *Proceedings of the International Symposium on Multiple-Valued Logic*, IEEE Computer Society Press, Victoria, BC, May 26-29, 1991, 238-245.

[3] Hähnle, R. *Automated Deduction in Multiple-Valued Logics.* International Series of Monographs on Computer Science, vol. 10. Oxford University Press, 1994.

[4] Hähnle, R. Many-valued logic and mixed integer programming. *Annals of Mathematics and Artificial Intelligence*, 12(3,4):231–264, Dec. 1994.

[5] Jackson, P., and Pais, J. Computing Prime Implicants. *Proceedings of the* 10^{th} *International Conference on Automated Deduction*, Kaiserslautern, W. Germany, July, 1990. In *Lecture Notes in Artificial Intelligence* (M. Stickel, ed.), Springer-Verlag, Vol. 449, 543-557.

[6] Jackson, P. Computing prime implicants incrementally. *Proceedings of the* 11^{th} *International Conference on Automated Deduction*, Saratoga Springs, NY, June, 1992. In *Lecture Notes in Artificial Intelligence*, (D. Kapur, ed.), Springer-Verlag, Vol. 607, 253-267.

[7] Kean, A., and Tsiknis, G. An incremental method for generating prime implicants/implicates. *Journal of Symbolic Computation* **9** (1990), 185-206.

[8] Kean, A., and Tsiknis, G. Assumption based reasoning and clause management systems. *Computational Intelligence* **8,1** (Nov. 1992), 1-24.

[9] Murray, N.V., and Rosenthal, E. Dissolution: Making paths vanish. *J.ACM* **40,3** (July 1993), 504-535.

[10] Murray, N.V., and Rosenthal, E. Resolution and path dissolution in multiple-valued logics. *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, Charlotte, NC, October 16-19, 1991. In *Lecture Notes in Artificial Intelligence*, (Z. Ras & M. Zemankova, eds.), Springer-Verlag, Vol. 542, 570-579.

[11] Murray, N.V., and Rosenthal, E. "Adapting Classical Inference Techniques to Multiple-Valued Logics Using Signed Formulas." *Fundamenta Informaticae* **21,3** (Sept. 1994), 237-253.

[12] Nelson, R.J., Simplest normal truth functions, *J. Symbolic Logic*, **20** (1955), 105-108.

[13] Ngair, T. A new algorithm for incremental prime implicate generation. *Proceedings of IJCAI-93*, Morgan Kaufmann, Chambery, France, August, 1993, 46-51.

[14] Ramesh, A., Becker, G., and Murray, N.V. "CNF and DNF Considered Harmful for Computing Prime Implicants/Implicates." To appear, *Journal of Automated Reasoning*.

[15] Reiter, R. and de Kleer, J. Foundations of assumption-based truth maintenance systems: preliminary report. *Proceedings of the* 6^{th} *National Conference on Artificial Intelligence*, Morgan Kaufmann, 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] Smith III W.R. Minimization of multivalued functions. In D.C Rine editor, *Computer Science and multiple-valued logic*, North Holland 1984, 227-267.

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

[19] Su S.Y.H and Cheung P.T., Computer simplification of multi-valued functions. In D.C Rine editor, *Computer Science and multiple-valued logic*, North Holland 1984, 195-226.