A Correct Optimized IPA

by
Alex Kean
and
George Tsiknis

Technical Report 92-12
June 1992

Department of Computer Science
University of British Columbia
Rm 333 - 6356 Agricultural Road
Vancouver, B.C.
CANADA V6T 1Z2

A Correct Optimized IPIA

ALEX KEAN and GEORGE TSIKNIS

Department of Computer Science, University of British Columbia,
Vancouver, British Columbia, Canada V6T 1Z2

Technical Report 92-12
5th June 1992

Email: kean@cs.ubc.ca, tsiknis@cs.ubc.ca

In response to the demands of some applications, we had developed an algorithm, called IPIA, to incrementally generate the prime implicates/implicants of a set of clauses. In an attempt to improve IPIA some optimizations were also presented. It was pointed out to us that some of these optimizations, namely the *subsumption* and *history restriction*, are self conflicting. Subsumption is a necessary operation to guarantee the primeness of implicates/implicants and *history restriction* is a scheme that exploits the history of consensus operation to avoid generating non prime implicant/implicates. The original IPIA, where *history restriction* was not consider, was proven correct. However, when *history restriction* was introduced later in the optimized version, it interacted with the subsumption operation to produce an incomplete set of prime implicants/implicates. This paper explains the problem in more details, proposes a solution and provides a proof of its correctness.

1 Introduction

The problem of computing prime implicates or implicants has been extensively studied in switching theory as part of the Boolean function minimization problem (Bartee *et al.*, 1962; Biswas, 1975; Kohavi, 1978; Tison, 1967). In the study of artificial intelligence an incremental method for generating prime implicates/implicants is appropriated for applications like the clause management system and assumption based reasoning (Kean and Tsiknis, 1993; Kean and Tsiknis, 1992).

Hereafter, we shall assume a propositional language with vocabulary $\mathcal{V}$; a set of logical connectives $\{\wedge, \vee, \neg, \rightarrow\}$; and sentences formed using only the vocabulary and the set of logical connectives. Additionally, a *literal* is either a or $\neg a$; a *disjunctive clause* is

a disjunction of literals; and a *conjunctive normal form* (CNF) formula is a conjunction of disjunctive clauses. A *disjunctive normal form* (DNF) formula is a disjunction of conjunctive clauses, in which a conjunctive clause is a conjunction of literals. We shall conveniently treat a clause as a set of literals and a formula as a set of clauses.

Informally, given a set of propositional sentences, a prime implicate of Σ is a *minimal* sentence entailed by Σ while a *minimal* sentence that entails Σ is a prime implicant of it. Traditionally, the notion of *implicant* is associated with formulas in disjunctive normal form while implicate is used with formula in conjunctive normal form. In these cases, they are formally defined as follows.

Given a conjunctive clause Q and a DNF formula $\mathcal{F}$, Q is an *implicant* of $\mathcal{F}$ if $\models Q \rightarrow \mathcal{F}$. Q is a *prime implicant* of $\mathcal{F}$ if Q is an implicant of $\mathcal{F}$ and there is no other implicant Q' of $\mathcal{F}$ such that $\models Q \rightarrow Q'$.

Conversely, given a disjunctive clause Q and a CNF formula $\mathcal{F}$, Q is an *implicate* of $\mathcal{F}$ if $\models \mathcal{F} \rightarrow Q$. Q is a *prime implicate* of $\mathcal{F}$ if Q is an implicate of $\mathcal{F}$ and there is no other implicate Q' of $\mathcal{F}$ such that $\models Q' \rightarrow Q$.

In this exposition we deal with sets of disjunctive clauses that is, formulae in conjunctive normal form, and implicates only. Since the notion of an implicate is a dual to that of an implicant, similar results can be obtained for the latter. For legibility, a clause is represented by the juxtaposition of its literals (eg. $x\overline{y}z$). If $M_1, M_2, \ldots, M_k$ are clauses, then for convenience the juxtaposition $M_1 M_2 \ldots M_k$ will represent the clause $\bigcup_{i=1}^{k} M_i$.

The problem of computing prime implicates incrementally is defined as follows: Given a set of clauses Σ (a formula), its corresponding set of prime implicates $PI(\Sigma)$ and a clause C, compute the set of prime implicates of $\Sigma \cup \{C\}$, or alternately the set $PI(\ PI(\Sigma) \cup \{C\}\ )$. There are two criteria for such an algorithm. First, the algorithm should not rely on canonical form[1] of the formula as most of the conventional methods do except those by Slagle *et al.* (1970) and Tison Method (1967). Second, the algorithm should exploit the properties of prime implicates so that the generation of prime implicates will be efficient.

An incremental method for generating prime implicates that satisfies the above two criteria was developed in (Kean and Tsiknis, 1990)[2]. The algorithm is called IPIA and is reproduced below for reference.

The algorithm works in stages. At each stage i, it performs consensus operations[3] between the new clauses generated at previous stages and existing clauses in Π, the input set of prime implicates, on the i-th biform variable of C that is, the literal $x_i \in C$ for which $\overline{x_i} \in P$ for $P \in \Pi$. It never performs consensus among existing clauses in Π and

[1]Let $\mathcal{S}$ be a set of clauses over a set of variables V. A clause $C \in \mathcal{S}$ is said to be in *canonical form* if every variable in V occurs in C.

[2]Familiarity with the definitions in (Kean and Tsiknis, 1990) is assumed.

[3]A consensus operation between two clauses is like a resolution operation between them with the exception that it is defined only if its resolvent is fundamental (i.e. non-tautological).

Algorithm: Incremental Prime Implicant/Implicate Algorithm(IPIA)

Input: A set of prime implicants Π of a formula $\mathcal{F}$ and a clause C.
Output: The set $\Sigma \cup \Pi$ is the set of prime implicants of $\Pi \cup \{C\}$.

Step 1.0 Initialize $\Sigma = \{C\}$. Delete any $D \in \Sigma \cup \Pi$ that is subsumed by another $D' \in \Sigma \cup \Pi$. **If** C is deleted then STOP.

Step 2.0 For each biform variable x occurring in C **do**

Step 2.1 For each $S \in \Sigma$ and $P \in \Pi$ such that S, P have consensus on x **do**
Step 2.1.1 $T = CS(S,P,x)$
Step 2.1.2 $\Sigma = \Sigma \cup T$.
end

Step 2.2 Delete any $D \in \Sigma \cup \Pi$ such that there is another $D' \in \Sigma \cup \Pi$ that subsumes D.
end
end

Algorithm 1.1: IPIA

among the newly generated clauses in Σ. Because of this pattern, the algorithmic process of IPIA can be represented by a tree, called a consensus tree or $CTree(\Pi, C)$. The root of $CTree(\Pi, C)$ is C and every other node n is a clause generated by a consensus operation between the parent of n, say m and the clause of Π that labels the edge (n, m).

The correctness of the IPIA algorithm, also shown in (Kean and Tsiknis, 1990) implied that the imposed restrictions described above on the consensus operations were justified. A further analysis of the problem revealed that some additional optimization is also possible. A number of optimizations were also examined in the same paper including the one known as *history restriction*. According to this, each clause S in Σ was assigned the set of biform literals of C on which consensus operations had been performed in order to generate S. Then a consensus operation between a clause S in Σ and a P in Π was not allowed if P was introducing biform literals already in the history of S. This optimization was aiming at prohibiting the generation of non-prime implicates.

Unfortunately, this restriction is too strong because not only it prohibits the generation of non-prime implicates, it also prohibits certain prime implicates from being generated. This error occurs when *subsumption* and *history restriction* interact in certain way. The error was first reported by Peter Jackson [4] and later by Johan de Kleer [5].

In Peter Jackson's example, $PI(\Sigma) = \{\overline{bdf}, b\overline{d}f\}$ and the input clause $C = \{be\overline{df}\}$.

[4] CADE90

[5] de Kleer's counter example was reported to us through electronic mail.

The *CTree* is shown in figure 1 where the history of a clause is displayed on the left of the clause, enclosed in "< >".

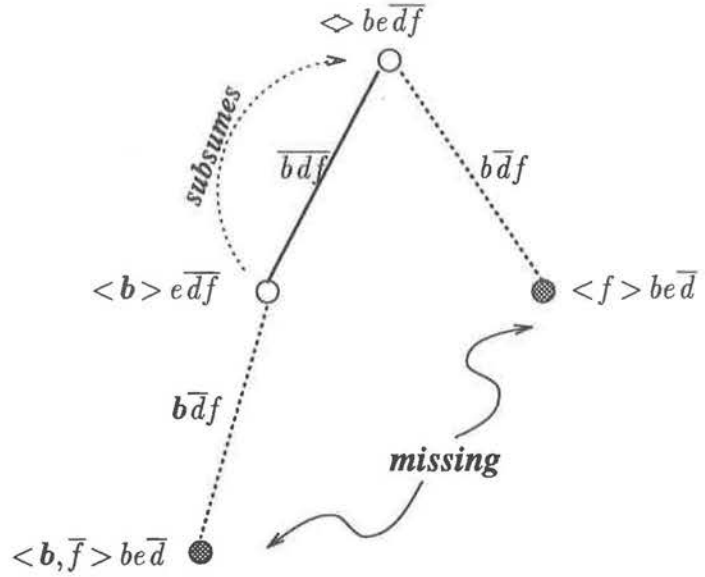

Figure 1: Jackson's Example

Starting from the root with an empty history, we first resolve on the biform literal b with the prime implicate $\overline{bdf}$ to yield $< \boldsymbol{b} > e\overline{df}$. At this point, it subsumes its parent and therefore the *right* branch (denoted by a dotted line) is not present. At the next stage, according to the history restriction the other prime implicate $b\overline{d}f$ is prohibited from resolving with $< \boldsymbol{b} > e\overline{df}$ on the biform literal $\overline{f}$ because $b\overline{d}f$ contains b which is in the history of $e\overline{df}$. Since there are no other possible consensus, the history restriction produces an incomplete set of prime implicates of $PI(\Sigma \cup \{C\})$, missing out $be\overline{d}$.

The second example, due to Johan de Kleer, has a similar structure but it is much simpler and is shown in figure 2.

The intuition behind the *history restriction* is that whenever a clause S is generated by a consensus operation that reintroduces biform literals of C that were previously resolved away, there is a great chance that this clause is not minimal. More specifically, it was believed that another clause S', generated in a way not violating the history restriction, must exists and subsumes S.

The proof of the existence of such a clause S' in the proof of lemma 7.1 of (Kean and Tsiknis, 1990) fell short in the sense that it did not take into account the case when such a consensus might not exists due to subsumption. In this paper, we propose a solution to the above reported error and prove its correctness. The solution is simply to separate the concept of a *history* and *restriction*. Thus, the *history* remains the same as in (Kean and Tsiknis, 1990) and the *restriction* changes according to subsumption. Hence, a consensus can be prohibited by the *restriction set* and not the *history*.

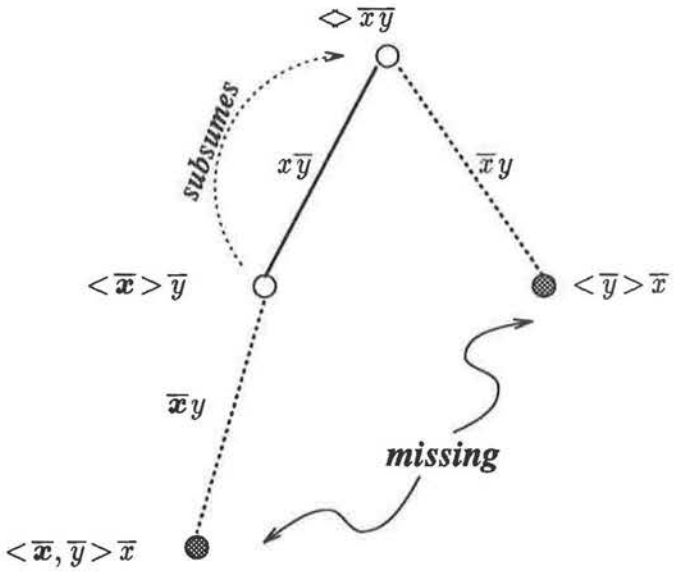

Figure 2: de Kleer's Example

2 Definitions

Once again, we assume the algorithm is applied to $\Pi \cup \{C\}$ and the biform part of C is denoted as $[C] = \{x_1, \ldots, x_n\}$. The correct and non-optimized version of IPIA (or simply IPIA) was presented in algorithm 1.1 as appeared in (Kean and Tsiknis, 1990, pp 190). The *history* and *restriction* definitions presented below have not been accommodated in the IPIA algorithm. These definitions are used to demonstrate some properties about non-minimality. When the result of these properties are derived, we shall integrate them into IPIA to derive a new corrected version called the *Corrected Optimized IPIA* (algorithm 4.1).

The definition of a history of a consensus remains the same as appeared in (Kean and Tsiknis, 1990) and is duplicated here for ease of reference.

Definition 2.1 (History) *For each clause $S \in \Sigma$ the history of S (history(S)) is defined as follows:*

a) $history(C) = \emptyset$

b) *If in step 2.1.1 of IPIA, $S = CS(S', P, x_i)$ for some $S' \in \Sigma$, $P \in \Pi$ and $x_i \in [C]$, then* $history(S) = history(S') \cup \{x_i\}$.

Thus, the *history* of a clause S contains all the biform literals of C that were involved in the chain of consensus operations that generates S. In addition to the notion of a history, we shall define a variant of it called a *restriction.*

Definition 2.2 (Restriction) *For each clause $S \in \Sigma$ the restriction of S (restriction(S)) is defined as follows:*

a) *restriction*$(C) = \emptyset$

b) *In step 2.1.1, if $S = CS(S', P, x_i)$ for some $S' \in \Sigma$, $P \in \Pi$ and $x_i \in [C]$, then set restriction*$(S) =$ *restriction*$(S') \cup \{x_i\}$ *and*

c) *In step 2.2, if D' is the clause that subsumes D then set restriction*$(D) =$ *restriction*$(D) \cap$ *restriction*(D').

Note that history and restriction are defined for the clauses in Σ, that is the clauses generated by chains of consensus operations starting at the input clause C. In order to have history and restriction defined for every clause, we set for every clause $P \in \Pi$, $history(P) = \emptyset$ and $restriction(P) = \emptyset$. The above settings are intuitively acceptable since elements of Π were given as input to the algorithm and no consensus operations using the root C were involved in generating them.

As a notation, if $history(S) = \mathcal{H}_S$ and $restriction(S) = \mathcal{R}_S$, we shall denote the clause S as $<\mathcal{H}_S, \mathcal{R}_S>S$. Thus, when using the algorithm IPIA, we shall compute the restriction for each clause generated and update its restriction if subsumption occurs. The restriction of a clause will be used to demonstrate that in any stage of the IPIA algorithm, if a clause S is generated and contains literals which are in the $restriction(S)$ that is, $S \cap restriction(S) \neq \emptyset$, the clause S is subsumed by another clause S' that is generated in the same stage and does not violate the restriction. This implies that clauses like S can be avoided if the algorithm prohibits consensus operation that introduces literals in the restriction of S.

Note that the notion of a restriction is similar to that of a history and the only difference is at step (c) of the restriction definition that changes (restricts) the restrictions of a clause whenever it subsumes another. The use of the intersection operation at this step is justified by the following observation. When a clause $<\mathcal{H}_S, \mathcal{R}_S>S$ subsumes $<\mathcal{H}_T, \mathcal{R}_T>T$ at step 2.2 of IPIA, T is deleted because the presence of S ensures that any implicate generated by T will be subsumed by an implicate generated by S. However, such assurance is not there when the mentioned restriction is enforced. if $\mathcal{R}_S$ and $\mathcal{R}_T$ are different sets there may exists consensus operation allowed by $\mathcal{R}_T$ but prohibited by $\mathcal{R}_S$. By updating $\mathcal{R}_S$ to become the intersection of the old $\mathcal{R}_S$ and $\mathcal{R}_T$, such assurance is reinstated.

Note that when S subsumes T, the operation of updating the restriction of S implies that the update is performed to *a particular* clause S. The reason is that there may be many clauses that subsume T but we only choose one such S. We shall define the notion of a *proxy* to facilitate the reference to such a clause.

Definition 2.3 (Proxy) *For any clause S generated by IPIA, we define a proxy of S as follows:*

1. *When $\langle \mathcal{H}_S, \mathcal{R}_S \rangle S$ is the clause T generated at step 2.1.1, the proxy of $\langle \mathcal{H}_S, \mathcal{R}_S \rangle S$ is itself;*
2. *If $\langle \mathcal{H}_S, \mathcal{R}_S \rangle S$ is the clause D at step 2.2, then the clause $\langle \mathcal{H}_{D'}, \mathcal{R}_{D'} \rangle D' \in \Sigma \cup \Pi$ in the same step is the proxy of $\langle \mathcal{H}_S, \mathcal{R}_S \rangle S$ and because of the subsumption and the intersection of restriction operations, the restriction $\mathcal{R}_{D'} \subseteq \mathcal{R}_S$.*
3. *If Q is the proxy of S and S is the proxy of T, then Q is the proxy of T. For the same reason, $\mathcal{R}_Q \subseteq \mathcal{R}_S \subseteq \mathcal{R}_T$.*

As a corollary to the definition, every clause in Σ has a proxy.

Corollary 2.1 *At any stage of IPIA, if S is a clause in Σ, there is a proxy of S in $\Sigma \cup \Pi$.*

Finally, a simple property of subsumption says that if A subsumes B and there is a consensus $CS(B, P, x)$, then using the same clause P and resolving on the same biform literal x, either A subsumes $CS(B, P, x)$ or the consensus $CS(A, P, x)$ subsumes $CS(B, P, x)$.

Lemma 2.1 *Let A, B and P be clauses and $CS(B, P, x)$ not be null*[6]*. If A subsumes B then either* (1) *A subsumes $CS(B, P, x)$ or* (2) *$CS(A, P, x)$ subsumes $CS(B, P, x)$.*

Proof : Let $B = xM_B$ and $P = \overline{x}M_P$. Assume that A subsumes B and there is a consensus $CS(B, P, x) = M_B M_P$.

1. If $x \notin A$, since $A \subseteq B$, $A \subseteq M_B M_P$. Therefore (i) holds.
2. If $x \in A$, since $A \subseteq B$, there is a consensus $CS(A, P, x) = M_A M_P$ such that $M_A \subseteq M_B$. Thus, (ii) holds. **QED**

We shall denote a chain of consensus operations

$$CS(\ldots CS(CS(C, P_1, x_1), P_2, x_2) \ldots, P_n, x_n)$$

by the generalized consensus $GCS(C, P_1, \ldots, P_n, < x_1, \ldots, x_n >)$ or without the biform variables as in $GCS(C, P_1, \ldots, P_n)$. Note that in the latter, the biform variables are left unspecified.

[6] If two clauses B and P have exactly one biform literal x then their consensus $CS(B, P, x)$ is defined (and it is their resolvent); otherwise it is null.

3 Restriction Optimization

Let the biform part of the input clause C be $[C] = \{x_1, \ldots, x_n\}$ and the C-variable order (the order in which the biform variables of C are selected at step 2.0 of IPIA) be $x_1, \ldots, x_n$. Therefore, for each $i, 1 \leq i \leq n$, stage i consists of all the consensus operations with respect to x_i that are performed by the algorithm.

The goal of this section is to prove that any clause generated by IPIA that violates the restriction condition is not minimal that is, another clause is also generated and subsumes the first. First, we need the following lemma.

Lemma 3.1 *Let* $<\mathcal{H}_Q,\mathcal{R}_Q>Q = GCS(C, P_{q_1}, \ldots, P_{q_m}, < x_{q_1}, \ldots, x_{q_m} >)$ *where* $P_{q_i} \in \Pi$ *and* $1 \leq q_1 < q_2 \cdots < q_m \leq n$. *Then at the end of stage* q_m *of IPIA, there is a clause* $<\mathcal{H}_T,\mathcal{R}_T>T \in \Sigma$ *such that* $<\mathcal{H}_T,\mathcal{R}_T>T$ *subsumes* $<\mathcal{H}_Q,\mathcal{R}_Q>Q$ *and* $\mathcal{R}_T \subseteq \{x_{q_1}, \ldots, x_{q_m}\}$.

Proof : By induction on the stages m.

When $m = 0$ then Q is the input clause $<\mathcal{H}_C,\mathcal{R}_C>C$. By the definition of restriction, $\mathcal{R}_C = \emptyset$, and $<\mathcal{H}_T,\mathcal{R}_T>T$ is either $<\mathcal{H}_C,\mathcal{R}_C>C$ or the proxy of $<\mathcal{H}_C,\mathcal{R}_C>C$. In either case $\mathcal{R}_T = \emptyset$ and the lemma is true.

Assume it is true for all stages $< m$ and we prove it for m. We assume that at the end of stage q_{m-1}, there is a clause $<\mathcal{H}_{T_{m-1}},\mathcal{R}_{T_{m-1}}>T_{m-1}$ that subsumes $Q_{m-1} = GCS(C, P_{q_1}, \ldots, P_{q_{m-1}}, < x_{q_1}, \ldots, x_{q_{m-1}} >)$ and $\mathcal{R}_{T_{m-1}} \subseteq \{x_{q_1}, \ldots, x_{q_{m-1}}\}$.

By the assumption, $Q = CS(Q_{m-1}, P_{q_m}, x_{q_m})$ and by lemma 2.1, either T_{m-1} or $CS(T_{m-1}, P_{q_m}, x_{q_m})$ subsumes Q. If T_{m-1} subsumes Q, the lemma is true because $\mathcal{R}_{T_{m-1}} \subseteq \{x_{q_1}, \ldots, x_{q_{m-1}}\}$. Otherwise, $<\mathcal{H}_{T_m},\mathcal{R}_{T_m}>T_m = CS(T_{m-1}, P_{q_m}, x_{q_m})$ subsumes Q. The following cases need to be considered.

1. If $<\mathcal{H}_{T_{m-1}},\mathcal{R}_{T_{m-1}}>T_{m-1} \in \Pi$, then $<\mathcal{H}_{T_m},\mathcal{R}_{T_m}>T_m = CS(T_{m-1}, P_{q_m}, x_q\, m)$ must also be in Π. Since by the definition of restriction, all clauses in $\mathcal{P}$ have empty history and restriction, $\mathcal{R}_{T_m} \subseteq \{x_{q_1}, \ldots, x_{q_m}\}$ and the lemma is true.

2. If $T_{m-1} \in \Sigma$ and $<\mathcal{H}_{T_m},\mathcal{R}_{T_m}>T_m$ is generated at stage q_m and since $\mathcal{R}_{T_m} = \mathcal{R}_{T_{m-1}} \cup \{x_{q_m}\}$, therefore $\mathcal{R}_{T_m} \subseteq \{x_{q_1}, \ldots, x_{q_m}\}$ and the lemma is true.

3. If $<\mathcal{H}_{T_m},\mathcal{R}_{T_m}>T_m$ is deleted, at the end of stage q_m there is a clause $<\mathcal{H}_T,\mathcal{R}_T>T$, the proxy of $<\mathcal{H}_{T_m},\mathcal{R}_{T_m}>T_m$, such that $<\mathcal{H}_T,\mathcal{R}_T>T$ subsumes $<\mathcal{H}_{T_m},\mathcal{R}_{T_m}>T_m$ and $\mathcal{R}_T \subseteq \mathcal{R}_{T_m}$. Therefore $<\mathcal{H}_T,\mathcal{R}_T>T$ subsumes $<\mathcal{H}_Q,\mathcal{R}_Q>Q$ and $\mathcal{R}_T \subseteq \{x_{q_1}, \ldots, x_{q_m}\}$ and the lemma is true. **QED**

Next, we show that at the end of any stage, any clause generated by IPIA will have empty intersection with its own restriction. Intuitively, this means that the clauses generated and remain at the end of each stage do not reintroduce biform literals that are in their restriction.

Lemma 3.2 (Corrected Lemma 7.1 of (Kean and Tsiknis, 1990))
At the end of each stage of IPIA, for any clause $<\mathcal{H}_S,\mathcal{R}_S>S \in \Sigma$, $\mathcal{R}_S \cap S = \emptyset$.

Proof : By induction on stage i.

When $i = 0$, the only clause in Σ is $<\mathcal{H}_C,\mathcal{R}_C>C$ and by the definition restriction, $\mathcal{R}_C = \emptyset$. Therefore the lemma is true.

Assume it is true for all stages $< i$ and assume that at stage i, a clause S exists such that $<\mathcal{H}_S,\mathcal{R}_S>S = CS(S_j, P, x_i)$ and $\mathcal{R}_S \cap S \neq \emptyset$ where $<\mathcal{H}_{S_j},\mathcal{R}_{S_j}>S_j$ is a clause generated prior to stage i. We also assume that $<\mathcal{H}_{S_j},\mathcal{R}_{S_j}>S_j$ is the generalized consensus of $<\mathcal{H}_C,\mathcal{R}_C>C$ and the set of prime implicates $\mathcal{P} = \{P_1, \ldots, P_j\}$ denoted by

$$<\mathcal{H}_{S_j},\mathcal{R}_{S_j}>S_j = GCS(C, P_1 = \overline{x}_{r_1} B_1 M_1, \ldots, P_j = \overline{x}_{r_j} B_j M_j, < x_{r_1}, \ldots, x_{r_j} >)$$

where, for each $k, 1 \leq r_k < i$, P_k resolves on the biform variable x_{r_k} at stage r_k, $B_k \subseteq [C]$ (positive biform literals) and $M_j \cap [C] = \emptyset$ (monoforms). Let $L_k = \{x_{r_k}, \ldots, x_{r_j}\}$ for each $k, 1 \leq k \leq j$. (i.e. the future yet to be resolved positive biform on stage r_k). Then, by the consensus definition

$$<\mathcal{H}_{S_j},\mathcal{R}_{S_j}>S_j = x_i, \ldots, x_n \bigcup_{k=1}^{j} (B_k - L_k) \bigcup_{k=1}^{j} M_k.$$

If the prime implicant $P = \overline{x}_i BM$, then $CS(S_j, P, x_i)$ is

$$<\mathcal{H}_S,\mathcal{R}_S>S = x_{i+1}, \ldots, x_n \bigcup B \bigcup_{k=1}^{j} (B_k - L_k) \bigcup M \bigcup_{k=1}^{j} M_k$$

and $\mathcal{H}_S = \mathcal{H}_{S_j} \cup \{x_i\}$, $\mathcal{R}_S = \mathcal{R}_{S_j} \cup \{x_i\}$.

We want to prove that if $\mathcal{R}_S \cap S \neq \emptyset$ then there is a clause $<\mathcal{H}_{T'},\mathcal{R}_{T'}>T'$ generated by IPIA at the end of stage i that subsumes S and has the property that $\mathcal{R}_{T'} \cap T' = \emptyset$. Notice that by the inductive hypothesis, since $j < i$, $\mathcal{R}_{S_j} \cap S_j = \emptyset$. Hence $\mathcal{R}_S \cap S \neq \emptyset$ iff $\mathcal{R}_S \cap B \neq \emptyset$, i.e. B reintroduces biform literals that are in the restriction of S.

Let $\mathcal{H}^* = \mathcal{H}_{S_j} - B = \{x_{q_1}, \ldots, x_{q_m}\}$. Obviously, $\mathcal{H}^* \subseteq \{x_1, \ldots, x_{i-1}\}$ and assume that these literals appear in $\mathcal{H}^*$ in the same order they got resolved in generating $<\mathcal{H}_{S_j},\mathcal{R}_{S_j}>S_j$ from $<\mathcal{H}_C,\mathcal{R}_C>C$. Let $\mathcal{P}' = \{P_{q_1}, \ldots, P_{q_m}\}$ be the corresponding subset of the prime implicates that were involved in the consensus operation. Note that $\mathcal{P}' \subseteq \mathcal{P}$.

By theorem 5.1 (the correctness of IPIA) and lemma 6.1 (every clause in Π is used exactly once in IPIA) of (Kean and Tsiknis, 1990), the fact that $<\mathcal{H}_{S_j},\mathcal{R}_{S_j}>S_j = GCS(C, P_1, \ldots, P_j)$ is defined, and $\{P_{q_1}, \ldots, P_{q_m}\} \subseteq \{P_1, \ldots, P_j\}$, the generalized consensus $<\mathcal{H}_Q,\mathcal{R}_Q>Q = GCS(C, P_{q_1}, \ldots, P_{q_m})$ is defined.

By lemma 3.1, at the end of stage q_m, there is a clause $<\mathcal{H}_T,\mathcal{R}_T>T$ such that $<\mathcal{H}_T,\mathcal{R}_T>T$ subsumes $<\mathcal{H}_Q,\mathcal{R}_Q>Q$ and $\mathcal{R}_T \subseteq \{x_{q_1}, \ldots, x_{q_m}\}$. Similarly, at the end of stage $i - 1$, there

is a clause $<\mathcal{H}_{T'},\mathcal{R}_{T'}>T'$ which is the proxy of $<\mathcal{H}_T,\mathcal{R}_T>T$, in which $\mathcal{R}_{T'} \subseteq \{x_{q_1},\ldots,x_{q_m}\}$ according to the definition of proxy (definition 2.3). Hence,

$$<\mathcal{H}_{T'},\mathcal{R}_{T'}>T' \subseteq <\mathcal{H}_T,\mathcal{R}_T>T \subseteq <\mathcal{H}_Q,\mathcal{R}_Q>Q \tag{1}$$

Now we want to prove that $Q \subseteq (S_j \cup B)$. First, let

$$<\mathcal{H}_Q,\mathcal{R}_Q>Q = x_i,\ldots,x_n \bigcup B \bigcup_{k=1}^{m}(B_{q_k} - \{x_{q_k},\ldots,x_{q_m}\}) \bigcup_{k=1}^{m} M_{q_k} \text{ and} \tag{2}$$

$$<\mathcal{H}_{S_j},\mathcal{R}_{S_j}>S_j = x_i,\ldots,x_n \bigcup_{k=1}^{j}(B_k - \{x_{r_k},\ldots,x_{r_j}\}) \bigcup_{k=1}^{j} M_k \tag{3}$$

a) Notice that each $B_{q_k}, 1 \leq k \leq m$ in (2) is identical to a B_{r_s} for some $s, 1 \leq s \leq j$ in (3) (i.e. each q_k in (1) is equal to some r_s in (2)). Therefore, in order to prove

$$\bigcup_{k=1}^{m}(B_{q_k} - \{x_{q_k},\ldots,x_{q_m}\}) \subseteq \bigcup_{k=1}^{j}(B_k - \{x_{r_k},\ldots,x_{r_j}\}) \cup B \tag{4}$$

it is sufficient to prove that for each $k, 1 \leq k \leq m$

$$(B_{q_k} - \{x_{q_k}, x_{q_{k+1}},\ldots,x_{q_m}\}) \subseteq (B_{q_k} - \{x_{q_k}, x_{q_k}+1,\ldots,x_j\}) \cup B. \tag{5}$$

Let,

$$x_z \in (B_{q_k} - \{x_{q_k}, x_{q_{k+1}},\ldots,x_{q_m}\}) \text{ and}$$
$$x_z \notin (B_{q_k} - \{x_{q_k}, x_{q_k}+1,\ldots,x_j\}) \text{ and } x_z \notin B.$$

Therefore,

$$x_z \in \{x_{q_k}, x_{q_k}+1,\ldots,x_j\} \text{ and} \tag{6}$$
$$x_z \notin \{x_{q_k}, x_{q_{k+1}},\ldots,x_{q_m}\}. \tag{7}$$

This means that x_z is not in S_j, i.e. it was resolved away by some P_z and it was not reintroduced by any one of $P_{z+1},\ldots,P_j$. Hence, $x_z \in H_{S_j}$ and since $x_z \notin B$, by the construction of $\mathcal{H}^*$, $x_z \in \mathcal{H}^*$ or simply

$$x_z \in \{x_{q_1},\ldots,x_{q_m}\}. \tag{8}$$

But then (6) and (8) imply that $x_z \in \{x_{q_k}, x_{q_{k+1}},\ldots,x_{q_m}\}$ which contradict (7). Therefore (4) and (5) hold.

b) It now remains to show that

$$\bigcup_{k=1}^{m} M_{q_k} \subseteq \bigcup_{k=1}^{j} M_k \tag{9}$$

Since $\{q_1, q_2, \ldots, q_m\} \subseteq \{1, 2, \ldots, k\}$, relation (9) holds.

Consequently, (4) and (9) imply that $Q \subseteq (S_j \cup B)$. So, according to (1)

$$<\mathcal{H}_{T'},\mathcal{R}_{T'}>T' \subseteq <\mathcal{H}_T,\mathcal{R}_T>T \subseteq <\mathcal{H}_Q,\mathcal{R}_Q>Q \subseteq (S_j \cup B).$$

Since T' is the proxy of T, we have

$$\mathcal{R}_{T'} \subseteq \{x_{q_1}, \ldots, x_{q_m}\} \tag{10}$$

and by the inductive hypothesis,

$$\mathcal{R}_{T'} \cap T' = \emptyset. \tag{11}$$

Then at stage i, the operation $CS(T', P, x_i)$ is defined. For otherwise T' and P must have more than one biform variables which implies (using (10)) that S_j and P have the same property, which contradict the hypothesis that $CS(S_j, P, x_i)$ is defined. Thus, the clause $<\mathcal{H}_{T'\cup\{x_i\}},\mathcal{R}_{T'\cup\{x_i\}}>T'' = CS(T', P, x_i)$ is generated. In this case, $(\mathcal{R}_{T'} \cup \{x_i\}) \cap T'' = \emptyset$ because $\mathcal{R}_{T'} \cap T' = \emptyset$ (from 11) and $\mathcal{R}_{T'} \subseteq \{x_{q_1}, \ldots, x_{q_m}\}$ (from 10). **QED**

As a consequence of the above lemma, only the proxy's restriction needs be updated even though there may be many other clauses that subsume a clause.

4 Optimized IPIA

The previous lemma suggests that clauses in Σ that violates the *restriction* condition need not be generated. Therefore, the optimized algorithm first must keep track of the restriction of each clause it generates and second, it must check if P and $\mathcal{R}_S$ have non-empty intersection before it generates the consensus of a clause $<\mathcal{H}_S,\mathcal{R}_S>S \in \Sigma$ and $P \in \Pi$. If so, the consensus should not be generated since it will be subsumed latter. On the other hand, if $\mathcal{R}_S \cap P = \emptyset$ the consensus is generated.

The new algorithm that accommodates this optimization together with the *root* and *subsumption* optimization as suggested in (Kean and Tsiknis, 1990) is shown in algorithm 4.1. Note that *history* is not used in the algorithm. Obviously this notion is not needed there; it was only used in the proof of the lemmas for which it was introduced.

In the sequence we briefly discuss the behavior of the new algorithm on the examples presented in the introduction.

Example 4.1 *In Peter Jackson's example,* $PI(\Sigma) = \{\overline{bdf}, b\overline{df}\}$ *and the input clause* $C = \{be\overline{df}\}$. *The set of biform literals of C and* Π *is* $[C] = \{b,f\}$.

Step 1: $\Sigma = \{<> be\overline{df}\}$, biform $= b$, $CS(<> be\overline{df}, \overline{bdf}, b) =< b > e\overline{df}$.

At this point, $< b > e\overline{df}$ subsumes $<> be\overline{df}$ and updating its restriction results in $<> e\overline{df}$.

Algorithm: Corrected Optimized IPIA

Input: A set of prime implicants Π of a formula $\mathcal{F}$ and a clause C.
Output: The set $\Sigma \cup \Pi$ is the set of prime implicants of $\Pi \cup \{C\}$.

Step 0.0 Delete any $D \in \Pi \cup \{C\}$ that is subsumed by another $D' \in \Pi \cup \{C\}$. If C is deleted, STOP.
Step 1.0 *(Root optimization)* **For** each $P \in \Pi$ **do**
Step 1.1 If $CS(C, P, x) = C'$ for some $x \in [C]$ and C' subsumes C **then** set $C = C'$ and delete any $P \in \Pi$ that is subsumed by C.
end
Step 2.0 Set $\Sigma = \{C\}$.
Step 3.0 For each biform literal $x \in [C]$ **do**
Step 3.1 Set $\Sigma\_Children = \emptyset$ and $\Pi_x = \{P \in \Pi \mid \overline{P} \cap C = \{x\}\}$
Step 3.2 For each clause S in Σ **do**
Step 3.2.1 If
$restriction(S) \cap P = \emptyset$ and $CS(S, P, x) = S'$ for some $P \in \Pi_x$ and S' subsumes S
then
delete S from Σ, $restriction(S') = restriction(S)$ and set $S\_Children = \{S'\}$
else
set $S\_Children = \{CS(S, P, x) \mid P \in \Pi_x$ and $restriction(S) \cap P = \emptyset\}$ and $restriction(S') = restriction(S) \cup \{x\} \forall S' \in S\_Children$.
end
Step 3.2.2 Delete any $D \in \Pi \cup S\_Children$ that is subsumed by another $D' \in \Pi \cup S\_Children$.
Step 3.2.3 Add $S\_Children$ to $\Sigma\_Children$.
end
Step 3.3 Check subsumption among the clauses in $\Sigma\_Children$ that have been generated by the same clause in Π_x. If a clause D is subsumed by D' then delete D and set $restriction(D') = restriction(D') \cap restriction(D)$.
Step 3.4 Add the remaining $\Sigma\_Children$ to Σ.
end
end

Algorithm 4.1: Corrected Optimized IPIA

Step 2) $\Sigma = \{<> e\overline{d}f\}$, biform $= f$, $CS(<> e\overline{d}f, b\overline{d}f, f) =< f > be\overline{d}$.

Since there is no more biform literals, the algorithm terminates with the correct set of prime implicates. Johan de Kleer's example is similar and is omitted.

5 Conclusions

An optimization to the algorithm for incrementally generating prime implicates has been presented. The optimization is simple in the sense that the additional information required to keep when a clause is generated (called the restriction of this clause) can be maintained by simple operation of finite set union and intersection throughout the life time of the clause. This additional information is used to avoid generating unnecessary clauses and subsequent subsumption operations on them. The new technique, by using the notion of restriction instead of that of history, avoids the problems discussed in the introduction.

As a final remark, we note that although the worst case complexity of the algorithm remains the same, the number of subsumption operations performed by the new algorithm is expected to be much lower on the average. The following example shows that in this case, the restriction strategy saves a fair bit of work by not generating implicates that are to be subsumed latter.

Example 5.1 *Let* $C = x_1x_2x_3x_4$ *and* $\Pi = \{\overline{x}_1x_4a,\ \overline{x}_2ac,\ \overline{x}_3b_1,\ \overline{x}_3b_2,\ \overline{x}_3b_3,\ \overline{x}_4x_1a\}$.

Figure 3 shows the *CTree* of $\Pi \cup \{C\}$. The set of biform literals is $[C] = \{x_1, x_2, x_3, x_4\}$ and the algorithm is executed according to that ordering. Initially, there is exactly one consensus between the root and the prime implicate $\overline{x_1}x_4a$ on the first biform literal x_1. At stage 2, $\Sigma = \{<> x_1x_2x_3x_4, < x_1 > x_2x_3x_4a\}$ and the only prime implicate that has the negative biform $\overline{x_2}$ is $\overline{x_2}ac$. The consensus between the root and the prime implicate is $CS(<> x_1x_2x_3x_4, \overline{x_2}ac, x_2) =< x_2 > x_1x_3x_4ac$ and the other consensus $CS(< x_1 > x_2x_3x_4a, \overline{x_2}ac, x_2) =< x_1x_2 > x_3x_4ac$. The latter consensus subsumes the first and hence the restriction is updated to become $< x_2 > x_3x_4ac$ as denoted in figure 3 with the labels **old** and **new**. After stage 3, the prime implicate $\overline{x_4}x_1a$ reintroduces the already resolved biform literal x_1. According to the restriction condition four of the nodes (denoted by the dotted lines and crossed nodes in the figure) are prohibited from consensus operations because x_1 is in their restriction. In fact, these prohibited consensus are repeated on the right branch of the tree. If we did not prohibit these four nodes from consensus, there will be duplicated consensus and if there are more consensus after stage x_4, the wasteful duplication can be enormous.

Acknowledgement: We are indebted to Peter Jackson and Johan de Kleer for reporting the error to us and many thanks to Johan de Kleer for providing helpful comments in helping us to derive a fix for the error. We are also very grateful to Alan Mackworth, Jane Mulligan and Paul Gilmore for their comments and support.

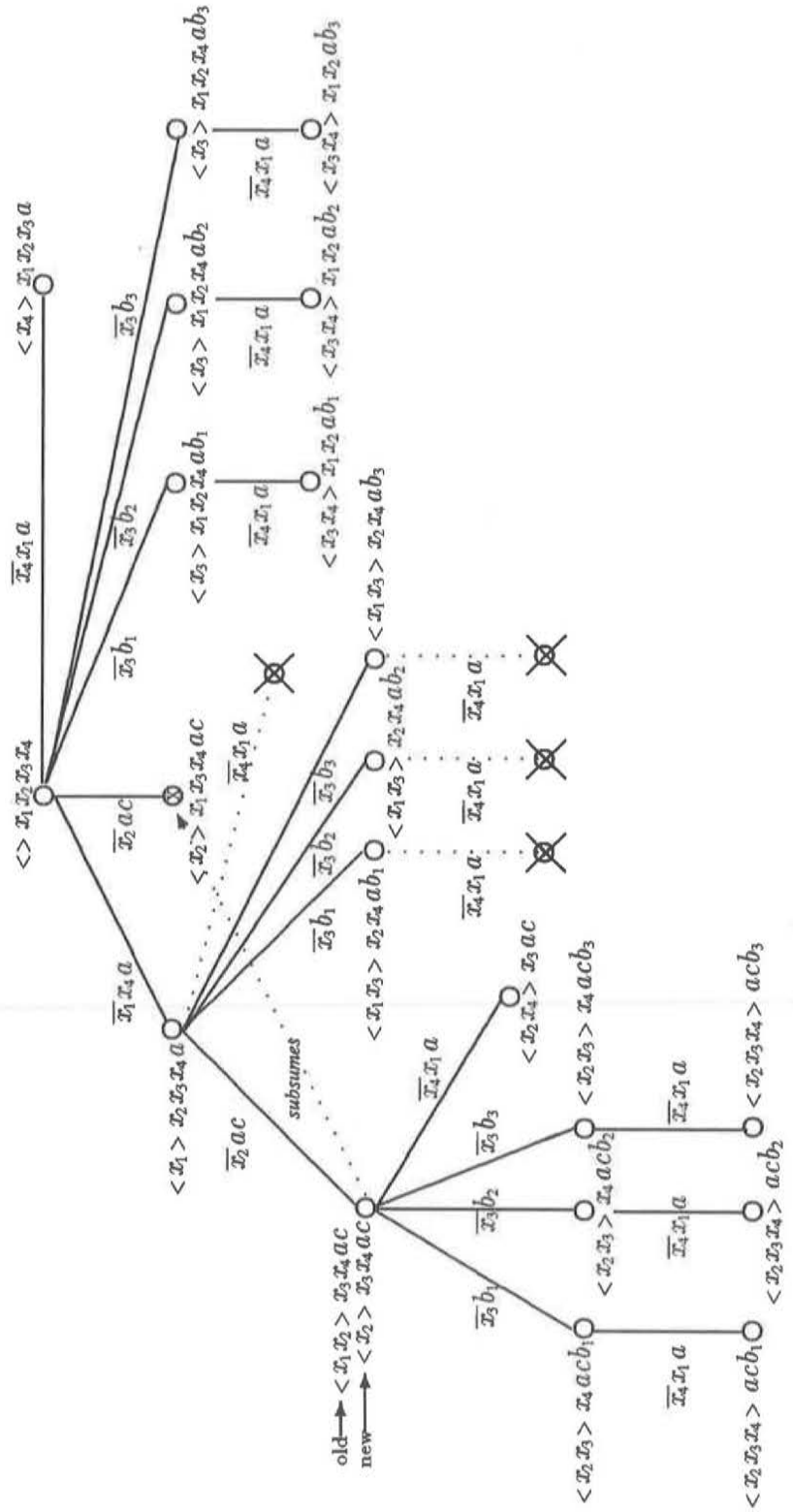

Figure 3: *CTree* for Example 5.1

References

[1] T.C. Bartee, I.L. Lebow, and I.S. Reed. *Theory and Design of Digital Machines.* McGraw-Hill Book, 1962.

[2] N.N. Biswas. *Introduction to Logic and Switching Theory.* Gordon and Breach Science, 1975.

[3] Alex Kean and George Tsiknis. An Incremental Method for Generating Prime Implicants/Implicates. *Journal of Symbolic Computation*, 9:185–206, 1990.

[4] Alex Kean and George Tsiknis. Assumption based Reasoning and Clause Management Systems. *Computational Intelligence*, 8(1):1–24, 1992.

[5] Alex Kean and George Tsiknis. Clause Management Systems. *To appear in the Computational Intelligence Journal*, 1993.

[6] Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill, second edition, 1978.

[7] J. R. Slagle, C. L. Chang, and R. C.T. Lee. A New Algorithm for Generating Prime Implicants. *IEEE Trans. on Computers*, 19(4), April 1970.

[8] P. Tison. Generalized Consensus Theory and Application to the Minimization of Boolean Functions. *IEEE Trans. on Computers*, 16(4):446–456, 1967.