CLAUSE MANAGEMENT SYSTEMS (CMS)

by

George Tsiknis and Alex Kean

Technical Report 88-21

October 1988

Clause Management Systems (CMS)

George Tsiknis and Alex Kean

Technical Report 88-21

October 1988

Department of Computer Science
The University of British Columbia
Vancouver, British Columbia
Canada V6T 1W5

Abstract

In this paper, we study the full extent of the *Clause Management System(CMS)* proposed by Reiter and de Kleer. The *CMS* is adapted specifically for aiding a reasoning system (*Reasoner*) in explanations generation. The *Reasoner* transmits propositional formulae representing its knowledges to the *CMS* and in return, the *Reasoner* can query the *CMS* for concise explanations w.r.t the *CMS* knowledge base. We argue that based on the type of tasks the *CMS* performs, it should represent its knowledge base Σ using a set of prime implicates $PI(\Sigma)$. The classification of implicates as prime, minimal, trivial and minimal trivial is carefully examined. Similarly, the notion of a support (or roughly, an explanation) for a clause including prime, minimal, trivial and minimal trivial is also elaborated. The methods to compute these supports from implicates and a preference ordering schema for the set of supports for a given clause are also presented. The generalization of the notion of a minimal support for a conjunction of clauses is also shown. Finally, two logic based diagnostic reasoning paradigms aided by the *CMS* are shown to exemplify the functionality of the *CMS* .

1 Introduction

In recent years, reasoning systems have rapidly evolved in sophistication and have penetrated almost every application domain. Such systems frequently have to cope with searching through huge knowledge bases in the attempt of finding explanations for a given query. The need for a supplementary system that can aid the reasoning systems to overcome this problem was recognized. For instance, there are the *Truth Maintenance System (TMS)* of Doyle (1979), the *Assumption-Based Truth Maintenance System (ATMS)* of de Kleer (1986) and the *Clause Management System (CMS)* of Reiter and de Kleer (1987).

One strategy to facilitate a reasoning system (or a *Reasoner*) is to have a supplementary system that maintains the *Reasoner* 's knowledge base KB, and in return, the *Reasoner* can query the supplementary system for explanations. The explanations to the *Reasoner* 's query computed by the supplementary system should be *concise* with respect to the knowledge base. The *Reasoner* can then use these explanations at its own discretion depending on the application domain. As a requirement, the supplementary system should be domain independent so that its functionality can be utilized in the wide range of domain dependent reasoning systems. There is considerable research in generating explanations including hypotheses generation method of Morgan (1971), reasoning in causes and effects by Cox and Pietryzkowski (1987), the THEORIST system of Poole *et al.* (1986) and etc.

In this paper, we study the full extent of the *Clause Management System (CMS)* proposed by Reiter and de Kleer (1987). In the *CMS* paradigm, A problem solving environment consists of a domain dependent *Reasoner* and a domain independent *CMS* . The *Reasoner* transmits a clause (it may be a 1st order formula) that describes some of its activities to the *CMS* . The *CMS* updates its knowledge base with this clause as a propositional clause (different atomic formulae correspond to different propositional variables). In the course of making a decision or performing a task, the *Reasoner* can query the *CMS* for explanations. The query consists of a propositional clause G, to which the *CMS* must respond with every *smallest* clause S such that $S \vee G$ but not S alone is a ***logical consequence*** of the clauses so far transmitted to the *CMS* . Such a clause is called the ***minimal support*** clause (or loosely speaking, the explanation) for G with respect to the *CMS* knowledge base.

In other words, the *CMS* informs the *Reasoner* that the negation $\overline{S}$ of every such S is a minimal hypothesis which, if known to the *Reasoner* , would sanction the conclusion G. The *Reasoner* can use these information to make a decision or choose the appropriate actions to perform. Thus, while the *Reasoner* 's actions depend on the domain of expertise, the *CMS* is highly domain independent. Figure 1 illustrates a possible architecture for a

Reasoner - CMS configuration.

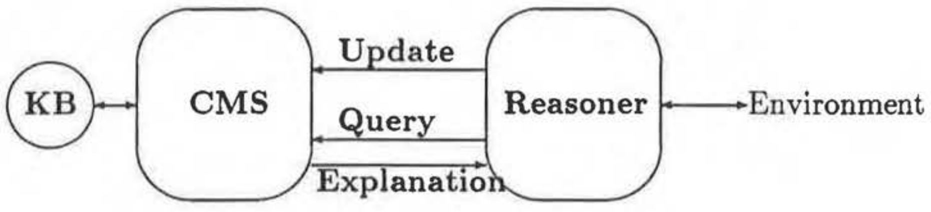


Figure 1: *Reasoner-CMS* Architecture

There are many applications using the *Reasoner-CMS* architecture. For example, Reiter and de Kleer (1987) describe how abductive reasoning can be accomplished in the *CMS* paradigm and how searching among alternatives in the search space can be facilitated by the *CMS* . In addition, de Kleer and Williams (1987) demonstrate the use of *Reasoner-ATMS* (a special kind of *CMS*) architecture in diagnostic reasoning. The full potential of the *Reasoner-CMS* architecture is still relatively unexplored. The set of possible configurations in the *Reasoner-CMS* architecture is rather interesting and research into their adequacy is far from complete. For instance, the tradeoff between the *CMS* maintaining the entire *Reasoner* 's KB *versus* the *Reasoner* maintaining its own KB and transmitting to the *CMS* only relevant knowledges with respect to the task it performs. Nevertheless, in this paper we will concentrate on the functionality of the *CMS* and its computational problems and leave such investigation as future work.

We will illustrate a possible *Reasoner-CMS* corporation schema by an example taken from (Reiter & de Kleer, 1987). Consider a reasoning system with knowledge base KB and assume that the *Reasoner* in its attempt to prove "g" has discovered that

$$KB \models p \land q \land r \rightarrow g$$
$$KB \models \neg p \land q \rightarrow g$$
$$KB \models \neg q \land r \rightarrow g.$$

Thus, the *Reasoner* transmits to the *CMS* the clauses "$\overline{p} \lor \overline{q} \lor \overline{r} \lor g$", "$p \lor \overline{q} \lor g$" and "$q \lor \overline{r} \lor g$". Suppose now that the *Reasoner* is interested in finding the minimal support for g. By querying the *CMS* with g it obtains the minimal support for g namely "$p \lor \overline{q}$", "$\overline{r}$" and "$\overline{g}$". This in turn implies that a minimal explanation[1] for "g" is either "$\overline{p} \land q$", "r" or "g" since $KB \models \overline{p} \land q \rightarrow g$, $KB \models r \rightarrow g$. $KB \models g \rightarrow g$.

[1]Notice that the notion of a support uses a disjunction and therefore, the negation of a support will

It has been shown that the the set of minimal supports $MS(G, \Sigma)$ for a query G can be computed trivially from the set of prime implicates $PI(\Sigma)$ of the *CMS* database Σ (Reiter & de Kleer, 1987). We shall present the full extent of the computation in section 4, 5 and 6. Since the set $PI(\Sigma)$ and Σ are logically equivalent, the *CMS* may choose to represent the set Σ as it is, the Simple-DB approach, or with extra effort and memory, compute and retain the set $PI(\Sigma)$ on-the-fly, the PI-DB approach. We shall briefly compare these two approaches.

Under the Simple-DB approach, the *CMS* stores the set of clauses transmitted by the *Reasoner* in its database without any alteration. Updating the *CMS* 's database Σ is trivially simple, i.e. $\Sigma = \Sigma \cup \{G\}$. Nevertheless the query processing is extremely expensive merely because the set $MS(G, \Sigma)$ and consequently the set $PI(\Sigma)$ must be computed for every different query G. Computing the set $PI(\Sigma)$ is most expensive but once the set $PI(\Sigma)$ is available, the set $MS(G, \Sigma)$ can be computed very efficiently by using special indexing and ordering schemes on $PI(\Sigma)$.

Naturally, the PI-DB approach is aimed at minimizing the expensive computation of the set $PI(\Sigma)$ by computing it incrementally (Reiter & de Kleer, pp 187, 1987). Thus under the PI-DB approach, the *CMS* stores the set of prime implicates, $\Pi = PI(\Sigma)$, of the clauses Σ it has received so far. When a new clause L is transmitted to the *CMS* , the *CMS* computes and stores $PI(\Pi \cup \{L\})$ using an incremental method discussed in (Kean & Tsiknis, 1988). As a consequence, the query processing for minimal support can be achieved very efficiently while updating the *CMS* database on the average is also relatively efficient using the incremental algorithm.

In the actual modelling of a *Reasoner-CMS* architecture, one must be cautious with the tradeoff between the Simple-DB and PI-DB approaches. If the *CMS* task is to perform vast numbers of updates, then the Simple-DB approach is superior simply because updates in Simple-DB approach take constant time. Conversely, if the *CMS* task is heavily related to query processing i.e., computing minimal supports, then the PI-DB approach is more suitable.

It is important to note that the size of the PI-DB database can be exponential, that is the number of prime implicates is potentially exponential (Chandra & Markowsky, 1978; Kean & Tsiknis, 1988). Consequently, the PI-DB approach potentially needs exponential space to store the prime implicates, but this is also the case for the Simple-DB approach each time a query is processed. The difference is simply that the Simple-DB approach does not retain

make it an implication. The choice of notation in using a disjunction in a support came from (Reiter & de Kleer, 1987) and in using implication in an explanation (or a cause) came from (Cox & Pietrzykowski, 1986).

the potentially exponential space after it is used but requires heavy recomputation for each query, while the PI-DB approach uses potentially exponential space but recomputation is kept to a minimum.

It is clear that both approaches require the same amount of space in the worst case. The remaining factor that discriminates between these two approaches lies in the time complexity of the query processing and update. In most reasoning environments, the turn-around time for the update can be compromised without affecting the *Reasoner* 's activities. Conceivably, the *Reasoner* issues an update and resumes its own activity without awaiting for the completion of the update. On the other hand, the query response time is crucial to the *Reasoner* . When a query is issued by the *Reasoner* , it must receive the response before it can continue its activity. In this scenario, one would prefer a fast response time to the query and compromise the turn-around time for update. Thus the PI-DB approach is preferable simply because it provides a faster query processing. In addition, augmenting the PI-DB approach with the incremental prime implicates generator (section 3) reduces the compromise for update to a minimum.

We shall begin by presenting the classification of implicates as minimal, prime, trivial and minimal trivial and their properties in section 2. On the issue of update, an incremental method for updating a set of prime implicates has been studied in (Kean & Tsiknis, 1988) and is briefly presented in section 3. The notion of a support and minimal support for a single clause, and the methods to compute them using minimal implicates is presented in section 4. The notion of a prime support is presented in section 4.1 and trivial support is explained in section 4.2. We elaborate further by showing that the set of minimal supports can be computed using only prime implicates. The actual algorithm to compute the set of minimal supports for a clause is given in section 5 and a preference ordering on the set of minimal supports is also discussed. Furthermore, the generalization of the notion of a support for a conjunction of clauses is studied in section 6.

Some extensive examples for two logic based diagnostic reasoning paradigms utilizing the *CMS* are presented to exemplify the functionality of the *CMS* in section 7. Finally, the conclusions and future work can be found in section 8.

2 Implicates

We shall begin with some definitions and notation that will be used throughout this paper. We shall assume a propositional language $\mathcal{L}$ with a vocabulary $\mathcal{V}$ containing a set of countably infinitely many propositional[2] variables and a set of logical connectives $\{\wedge, \vee, \neg\}$. A *variable* is denoted by a lowercase letter possibly subscripted (eg. $x, y, \ldots$) and a *literal* is a positive variable x or a negative variable $\overline{x}$ (or $\neg x$). We also call x and $\overline{x}$ a pair of *complementary* literals. We shall use the logical entailment relation, $\models$, with its standard logical definition and $Th(\Sigma)$ for the set of all the logical consequences of Σ, where $\Sigma \subseteq \mathcal{L}$.

A *clause*, denoted by an uppercase letter possibly with subscript (e.g. $A, B, \ldots$), is a disjunction of literals without repetition. The disjunction of literals is denoted by the juxtaposition the literals (e.g. $x\overline{y}z$) and for simplicity, a clause is also assumed to be the set of its literals. An *empty clause* (false) is denoted by the symbol $\Box$. A formula is a conjunction of clauses without repetition (CNF and the dual is DNF) and for simplicity, a formula is also a set of clauses denoted by capital Greek letters, possibly subscripted (e.g., $\Sigma, \Pi, \ldots$). For convenience, we will perform set operations i.e., union ($\cup$), intersection ($\cap$) and difference ($-$) on clauses and formulae and assume $A \cup B$ to mean the same as $A \vee B$.

Definition 2.1 (Subsumption) *A clause A is said to* <u>*subsume*</u> *another clause B if $A \subseteq B$.*

Given a set of clauses Σ, the function $SUB(\Sigma)$ is a subset of Σ such that for every clause $C \in SUB(\Sigma)$, there is no clause $C' \in \Sigma$ and $C' \neq C$ that subsumes C. The function $SUB(\Sigma)$ is used when subsumption is required on a set of clauses.

Definition 2.2 (Fundamental) *A clause C is a* <u>*fundamental*</u> *clause if C does not contain a complementary pair of literals, otherwise C is* <u>*non-fundamental.*</u>[3]

Definition 2.3 (Minimal/Prime Implicate) *Given a set of clauses Σ and a clause P,*

1. *P is an* <u>*implicate*</u> *of Σ if $\Sigma \models P$.*
2. *P is a* <u>*minimal implicate*</u> *of Σ if P is an implicate of Σ and there is no other implicate P' of Σ such that P' subsumes P.*
3. *P is a* <u>*prime implicate*</u>[4] *of Σ if P is an implicate of Σ and there is no other implicate P' of Σ such that $\models P' \rightarrow P$.*

[2]Since we are dealing specifically with propositional logic, the adjective "propositional" with be dropped whenever no ambiguity arises.

[3]Note that by the definition of a clause in this paper, a non-fundamental clause is a tautology.

[4]The notion of a prime implicate (the dual of prime implicant) is first introduced in [Slagle, Chang & Lee'70].

According to definition 2.3, the difference between a minimal implicate and a prime implicate is that the minimality relation, $\rightarrow$, for prime implicates excludes non-fundamental clauses whereas the definition of minimal implicate does not.

Example 2.1 *Let* $\Sigma = \{\overline{a}b, \overline{b}c, \overline{c}d, ae, \overline{b}e\}$ *and the vocabulary* $\mathcal{V} = \{a, b, c, d, e, f, g, \ldots\}$. *The set of minimal implicates are*

$$\{\overline{a}b, \overline{b}c, \overline{c}d, \overline{a}c, \overline{a}d, \overline{b}d, e, a\overline{a}, b\overline{b}, c\overline{c}, d\overline{d}, f\overline{f}, g\overline{g}, \ldots\}$$

and the set of prime implicates are $\{\overline{a}b, \overline{b}c, \overline{c}d, \overline{a}c, \overline{a}d, \overline{b}d, e\}$.

Notice that the set prime implicates is finite if the set Σ is so. According to the definition of implicate, a non-fundamental clause P is always an implicate of any set Σ. We shall distinguish such an implicate by the following definition.

Definition 2.4 (Trivial Implicate) *A non-fundamental clause P is a trivial implicate of any set Σ. A clause P is a minimal trivial implicate of Σ if P is both a minimal and a trivial implicate of Σ.*

In example 2.1, any clause of $\mathcal{L}$ that contains at least one pair of complementary literals is a trivial implicate of Σ but only $a\overline{a}, b\overline{b}, c\overline{c}, d\overline{d}, f\overline{f} \ldots$ (note that $e\overline{e}$ is not minimal) are minimal trivial implicates of Σ. Note that the set of minimal trivial implicates can be infinite if $\mathcal{V}$ is infinite. The distinction between these different types of implicates is crucial in the understanding of the notion of a support later.

Notation (Sets of Implicates) If Σ is a set of clauses, $I(\Sigma)$, $MI(\Sigma)$, $PI(\Sigma)$, $TI(\Sigma)$ and $MTI(\Sigma)$ denote the set of all implicates, minimal implicates, prime implicates, trivial implicates and minimal trivial implicates of Σ respectively.

Naturally, the set $I(\Sigma)$ is the set of theorems, $Th(\Sigma)$, of Σ. Thus, if Σ is inconsistent, $I(\Sigma) = \mathcal{L}$ and $\Sigma \models \Box$, hence $MI(\Sigma) = \{\Box\}$ and $PI(\Sigma) = \{\Box\}$ because the empty clause subsumes every other implicates. The set $MTI(\Sigma) = \emptyset$ because the empty clause is fundamental. If Σ is empty, then the set of all implicates $I(\Sigma)$ are all the tautologies of $\mathcal{L}$, i.e. $I(\Sigma) = TI(\Sigma)$. Consequently, $PI(\Sigma) = \emptyset$ and $MI(\Sigma) = MTI(\Sigma)$.

Recall that only fundamental clauses can be prime implicates. In addition, the set $MTI(\Sigma)$ contains only non-fundamental clauses. Naturally, the various sets of implicates defined here are closely related. The following lemmata and theorems illustrate the relations between the sets $PI(\Sigma)$, $MI(\Sigma)$ and $MTI(\Sigma)$ of a set of clauses Σ. First, we show that all prime implicates are fundamental.

Lemma 2.1 (PI is Fundamental) *Every prime implicate of a set of clauses Σ is fundamental.*

Proof : Let $P \in PI(\Sigma)$ and assume that P is non-fundamental. Then for any implicate G of Σ, $\models G \rightarrow P$ which contradicts the fact that P is a prime implicate of Σ. $\square$

Following the lemma, the set of $PI(\Sigma)$ contains only fundamental clauses. Since the set $MTI(\Sigma)$ contains only non-fundamental clauses, the following corollary follows immediately.

Corollary 2.1 *For any set of clauses Σ, $PI(\Sigma) \cap MTI(\Sigma) = \emptyset$.*

Recall that the difference between $PI(\Sigma)$ and $MI(\Sigma)$ is the minimality relation. In fact, the minimality relation ($\rightarrow$) expressed in $PI(\Sigma)$ is similar to the minimality ($\subseteq$) in $MI(\Sigma)$ if all clauses are fundamental. The following lemma states the relation between $PI(\Sigma)$ and $MI(\Sigma)$.

Lemma 2.2 *For any set of clauses Σ, $PI(\Sigma) \subseteq MI(\Sigma)$.*

Proof : Let $P \in PI(\Sigma)$ and assume that $P \notin MI(\Sigma)$. Then there exists an implicate Q of Σ such that Q subsumes P. But then $\models Q \rightarrow P$ which contradicts $P \in PI(\Sigma)$. $\square$

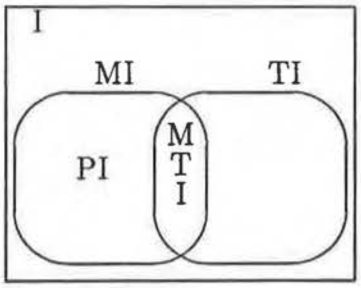
The motiviation for investigating the relations among $MI(\Sigma)$, $PI(\Sigma)$ and $MTI(\Sigma)$ can now be expressed in the following theorem.

Theorem 2.1 *For any set of clauses Σ, $MI(\Sigma) = PI(\Sigma) \cup MTI(\Sigma)$.*

Proof : If $P \in PI(\Sigma) \cup MTI(\Sigma)$, by lemma 2.2 and definition 2.4, $P \in MI(\Sigma)$. Conversely, assume that $P \in MI(\Sigma)$. If P is fundamental then there is no other implicate Q of Σ for which $\models Q \rightarrow P$, since the existence of such Q implies that Q subsumes P contradicting P being minimal. Therefore $P \in PI(\Sigma)$. If P is non-fundamental then by definition 2.4, $P \in MTI(\Sigma)$. Consequently, $P \in PI(\Sigma) \cup MTI(\Sigma)$. $\square$

The inclusion relations between the defined sets of implicates with respect to a set of clauses Σ are depicted by figure 2.

Figure 2 illustrates that the set $MI(\Sigma)$ is being partitioned into two disjoint sets $PI(\Sigma)$ and $MTI(\Sigma)$. Since finite set of Σ implies finite set of $PI(\Sigma)$, therefore $MI(\Sigma)$ is finite iff $MTI(\Sigma)$ is finite or $\mathcal{V}$ is finite. The investigation into the inclusion relations among these sets reveals that the set $MI(\Sigma)$ can be constructed given only the set $PI(\Sigma)$ because the set $MTI(\Sigma)$ can be constructed on-the-fly.



I	= Implicates
MI	= Minimal Implicates
TI	= Trivial Implicates
PI	= Prime Implicates
MTI	= Minimal Trivial Implicates

Figure 2: Implicate Inclusion Relations

Theorem 2.2 (Set of MTI) *For any set of clauses Σ over a language $\mathcal{L}$ with the vocabulary $\mathcal{V}$, $MTI(\Sigma) = \{x\overline{x} \mid x \in \mathcal{V}$ and no $P \in PI(\Sigma)$ subsumes $x\overline{x}\}$.*

Proof : Let $\mathcal{T} = \{x\overline{x} \mid x \in \mathcal{V}$ *and no* $T \in PI(\Sigma)$ *subsumes* $x\overline{x}\}$. If $T \in MTI(\Sigma)$ then T is of the form $x\overline{x}$, for if $T = x\overline{x}A$ for any clause A, then there is always a trivial implicate $x\overline{x}$ of Σ that subsumes T. Therefore T has the form $x\overline{x}$ where $x \in \mathcal{V}$ and no other implicate (nor prime implicate) of Σ subsumes it. Consequently, $T \in \mathcal{T}$. Conversely, we assume that $T \in \mathcal{T}$. T is a trivial implicate in the form $x\overline{x}$ and T is not minimal iff there exist a minimal implicate Q that subsumes T. In this case, Q is either x or $\overline{x}$ or $\Box$. In either case, Q is fundamental and, by the lemma 2.1, $Q \in PI(\Sigma)$. Therefore $T \in MTI(\Sigma)$. $\Box$

We will illustrate the formation of the set $MI(\Sigma)$ by assuming the set $PI(\Sigma)$ and construct the set $MTI(\Sigma)$ using $PI(\Sigma)$ by the following example.

Example 2.2 *Let* $\Sigma = \{pqr, \overline{p}q, \overline{q}r, \overline{p}q\overline{s}, \overline{r}\overline{s}, \overline{r}pt\}$,

then $PI(\Sigma) = \{r, \overline{s}, pt, qt, \overline{p}q\}$,
$MTI(\Sigma) = \{x\overline{x} \mid x \in \mathcal{V} - \{r, s\}\}$
and $MI(\Sigma) = PI(\Sigma) \cup MTI(\Sigma)$.

The notion of a logical entailment of a clause by a set of clauses Σ is expressed using the sets $MI(\Sigma)$ and $PI(\Sigma)$ by the following theorem.

Theorem 2.3 (Entailment) *Let Σ be a set of clauses,*

1. *if G is a clause, then $\Sigma \models G$ iff there is an $M \in MI(\Sigma)$ such that M subsumes G.*
2. *if G is a fundamental clause, then $\Sigma \models G$ iff there is a $P \in PI(\Sigma)$ such that P subsumes G.*

Proof : Let Σ be a set of clauses and G be a clause.

1. Assume $\Sigma \models G$. By definition 2.3, G is an implicate of Σ. If there is no $M \subset G$ such that $\Sigma \models M$, then by the definition of minimal implicate (definition 2.3), $G \in MI(\Sigma)$ and G subsumes itself. Otherwise, by the definition of $MI(\Sigma)$, there is an $M \in MI(\Sigma)$ such that M subsumes G. Conversly, assume that there is a $M \in MI(\Sigma)$ such that M subsumes G. By definition 2.3, $\Sigma \models M$. Since $M \subseteq G$, $\Sigma \models G$.

2. Similar argument as above except that since G is fundamental, any proper subset P of G is also fundamental. Therefore, by the definition of prime implicate (definition 2.3), $P \in PI(\Sigma)$. □

Consequently, the sets Σ, $MI(\Sigma)$ and $PI(\Sigma)$ are logically equivalent as expressed by the following theorem.

Theorem 2.4 (Logical Equivalence) *Suppose Σ is a set of clauses, then Σ, $MI(\Sigma)$ and $PI(\Sigma)$ are all logically equivalent in the sense that if a clause C is in one of the above sets, then the others logically entail C.*

Proof : A direct consequence of definition 2.3 and theorem 2.3. □

As a consequence of theorems 2.1, 2.2 and 2.4, the most important set of implicates of Σ is the set of prime implicates $PI(\Sigma)$. In fact, the set of prime implicates of Σ plays the central role in the *Clause Management System* presented here and will be used extensively throughout the paper.

3 Update

The update problem in the PI-DB approach can be formulated as computing the set of prime implicates $PI(\Pi \cup \{C\})$, where Π is the current *CMS* database (a set of prime implicates) and C is a new clause transmitted by the *Reasoner* to the *CMS* (Reiter & de Kleer, pp 187, 1987). Methods for generating prime implicates (the dual of implicants) from Boolean expressions have been studied extensively in the area of switching theory (Bartee *et al.*, 1962; Hwa, 1974; Slagle *et al.*, 1970). It is obvious that all of the conventional methods for generating prime implicates are applicable to the *CMS* update problem. Nevertheless they are inefficient in this formulation because they do not exploit the fact that Π is already a set of prime implicates. Intuitively, the property of implicates being prime implies that any effort to generate new prime implicates from a set of prime implicates Π will only yield Π itself i.e., $PI(\Pi) = \Pi$.

We shall briefly present the method which generates prime implicates incrementally. Some definitions used in the algorithm are presented below however, a more detailed presentation of the algorithm can be found in (Kean & Tsiknis, 1988). The algorithm is based on an idea similar to Tison's method (1967), adapted for the purpose of incrementally generating prime implicates. Given a set of clauses $\Sigma = F_1 \wedge \ldots \wedge F_n$, a variable x is a *biform* variable in Σ if $x \in F_i$ and $\overline{x} \in F_j$ for some $1 \leq i, j \leq n$. Also, given two clauses $A = xA'$ and $B = \overline{x}B'$, the *consensus* of A and B with respect to the biform variable x is $CS(A, B, x) = A'B'$ iff $A'B'$ is fundamental.

Incremental Prime Implicate Algorithm(IPIA)

```
Input: A set of prime implicates $\Pi$ of a set of clauses $\Sigma$ and a clause $C$.
Output: The set $\Omega \cup \Pi$ is the set $PI(\Pi \cup \{C\})$.
Step 1.0 Initialize $\Omega = \{C\}$. Delete any $D \in \Omega \cup \Pi$ such that there is another $D' \in \Omega \cup \Pi$ that
    subsumes $D$. If $C$ is subsumed then STOP.
Step 2.0 For each biform variable $x$ occurring in $C$ do
    Step 2.1 For each $S \in \Omega$ 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 $\Omega = \Omega \cup T$.
    Step 2.2 Delete any $D \in \Omega \cup \Pi$ such that there is another $D' \in \Omega \cup \Pi$ that subsumes $D$.
```

Intuitively, any biform variable of $\Pi \cup C$ that does not occur in C must occur in some $P \in \Pi$, but since Π is a set of prime implicates, the consensus among them cannot yield any new prime implicates. Moreover, we need to consider only the consensus of a clause from Ω and a clause from Π but never two clauses from the same set Ω or the set Π in Step 2.1. Intuitively, any possible consensus of two clauses that comes from the same set Ω or Π is subsumed by a clause in $\Omega \cup \Pi$ after completion of the algorithm. The correctness proof of the algorithm can be found in (Kean & Tsiknis, 1988). We shall demonstrate the incremental algorithm by the following example.

Example 3.1 Let $\Pi = \{\overline{t}x\overline{y}, a\overline{b}c, \overline{a}b\overline{c}\}$ and $C = a\overline{c}t$ a new clause. There are three iterations in the algorithm corresponding to the biform variables "a", "$\overline{c}$" and "t", and the consensus with respect to each biform variable are shown below respectively.

1) $\Omega = \{a\overline{c}t\}$, biform $= a$
   $T = CS(a\overline{c}t, \overline{a}b\overline{c}, a) = b\overline{c}t$
2) $\Omega = \{a\overline{c}t, b\overline{c}t\}$, biform $= \overline{c}$
   $T = CS(a\overline{c}t, a\overline{b}c, a) = a\overline{b}t$
3) $\Omega = \{a\overline{c}t, b\overline{c}t, a\overline{b}t\}$, biform $= t$
   $T_1 = CS(a\overline{c}t, \overline{t}x\overline{y}, t) = a\overline{c}x\overline{y}$

$$T_2 = CS(b\overline{c}t, \overline{t}x\overline{y}, t) = b\overline{c}x\overline{y}$$
$$T_3 = CS(a\overline{b}t, \overline{t}x\overline{y}, t) = a\overline{b}x\overline{y}$$
$$\Omega = \{a\overline{c}t, b\overline{c}t, a\overline{b}t, a\overline{c}x\overline{y}, b\overline{c}x\overline{y}, a\overline{b}x\overline{y}\}$$

The set $[\Omega \cup \Pi = \{a\overline{c}t, b\overline{c}t, a\overline{b}t, a\overline{c}x\overline{y}, b\overline{c}x\overline{y}, a\overline{b}x\overline{y}, \overline{t}x\overline{y}, a\overline{b}c, \overline{a}b\overline{c}\}$ after completion is the set of prime implicates of $PI(\Pi \cup \{C\})$.

4 Support

In this section, we shall present the notion of a support for a single clause (cf. Reiter & de Kleer, 1987) and postpone the discussion of the support for a conjunction of clauses until section 6.

Definition 4.1 (Support) *Let Σ be a set of clauses and G be a clause. A clause S is a support for G with respect to Σ if*

1. $\Sigma \models S \vee G$ *(or* $\Sigma \models \overline{S} \rightarrow G$*)*
2. $\Sigma \not\models S$.

A clause S is a minimal support for G w.r.t Σ if S is a support for G and there is no other support S' for G such that S' subsumes S.

The intuition behind a support clause S is that $\overline{S}$ is a hypothesis that implies the conclusion G pertaining to the knowledge base Σ. Further, $\overline{S}$ is not inconsistent with Σ otherwise Σ would imply any conclusion whatsoever. A minimal support clause is the smallest such hypothesis that implies G. We shall illustrate the notion of a support and minimal support using the following example.

Example 4.1 Assume that the *CMS* received the following set of clauses $\Sigma = \{p \rightarrow ab, b \rightarrow c, p \rightarrow ad\}$. If "$c$" is the query, then $\{\overline{c}, \overline{p}a, \overline{b}\}$ is the set of *minimal supports* for "c". This is obviously true because every clause in this set is consistent with Σ and $\Sigma \models c \rightarrow c$, $\Sigma \models p \wedge \overline{a} \rightarrow c$ and $\Sigma \models b \rightarrow c$.

We have claimed that the set of minimal supports for a clause G w.r.t. Σ can be easily computed from the set of minimal implicates of Σ. The following lemma and theorem on minimal supports and sets of minimal supports are similar to Reiter and de Kleer (1987) and are given here without proofs. The reader should note that our notion of *minimal implicate* corresponds to the notion of *prime implicants* in (Reiter & de Kleer, 1987). We decided to adopt the terminology used in switching theory instead, because *primeness* is a much stronger property than minimality.

Lemma 4.1 (Minimal Support) *Suppose Σ is a set of clauses, $MI(\Sigma)$ is the set of minimal implicates of Σ and G is a clause. If the clause S is a minimal support clause for G with respect to Σ, then there is an $M \in MI(\Sigma)$ such that $M \cap G \neq \emptyset$ and $S = M - G$.*

Let Σ denote a set of clauses, $MI(\Sigma)$ denote the set of minimal implicates of Σ, and G be a clause, and let $\Delta = \{M - G \mid M \in MI(\Sigma) \text{ and } M \cap G \neq \emptyset\}$. The above lemma ensures that if S is a minimal support for G, then S can be found in the set Δ. Unfortunately, the converse of the lemma is not true in the sense that there are members of Δ that are not minimal supports for G. This is because subsumption might occur among members of Δ. Consequently, subsumption must be considered in the construction of the set as the following theorem indicates.

Theorem 4.1 (Set of Minimal Supports) *Let Σ denote a set of clauses, $MI(\Sigma)$ denote the set of minimal implicates of Σ and G be a clause. The set*

$$MS(G, \Sigma) = SUB(\{M - G \mid M \in MI(\Sigma) \text{ and } M \cap G \neq \emptyset\})$$

is the set of minimal supports for G with respect to Σ.

There is a special case where the inverse of lemma 4.1 also holds in the sense that every support in Δ is also minimal. This is the case when the query is a clause with a single literal (*unit clause*) as stated in the following corollary.

Corollary 4.1 *Suppose Σ is a set of clauses, $MI(\Sigma)$ is the set of minimal implicates of Σ and $G = l$ is a unit clause. Then clause S is a minimal support clause for G with respect to Σ iff there is a $M \in MI(\Sigma)$ such that $l \in M$ and $S = M - l$.*

Note that if Σ is inconsistent, there is no support for G. This follows because the only clause in $MI(\Sigma)$ is the empty clause. Since the empty clause does not have any intersection with any clause G, $MS(G, \Sigma) = \emptyset$. Similarly, if $\Sigma \cup G$ is inconsistent (even though Σ is consistent), G has no support. On the other hand, if Σ is consistent and $\Sigma \models G$, by theorem 2.3, there is an $M \in MI(\Sigma)$ that subsumes G. Thus the only minimal support for G is $M - G = \{\Box\}$. For the same reason, if the query G is a non-fundamental clause (tautology), then the only minimal support clause for G is the empty clause.

4.1 Prime Support

Recall in example 4.1 that "$\overline{c}$" was a minimal support for "c". In fact, "$\overline{c}$" trivially supports "c" in the sense that "$\overline{c}c$" is a tautology. We shall distinguish this type of trivial support and elaborate on their role in section 4.2, but we shall first discuss the type of supports that are nontrivial in the following sense.

Definition 4.2 (Prime Support) *Let Σ be a set of clauses and G be a clause. A clause S is a fundamental support for G with respect to Σ if S is a support for G and $S \cup G$ is fundamental. A clause S is a prime support for G if S is both a minimal and fundamental support for G.*

Fundamental and prime supports are more restricted notions of their counterparts defined in the previous section, in the sense that they have to preserve an additional property namely, $S \cup G$ being fundamental. Consequently given a clause G, any support S for G that makes $S \cup G$ non-fundamental (called *trivial support* in section 4.2) is neither a fundamental nor prime support for G. Not surprisingly, the set of prime supports for a clause G w.r.t. Σ can be easily computed from the set of prime implicates of Σ.

Theorem 4.2 (Set of Prime Supports) *Suppose Σ is a set of clauses, $PI(\Sigma)$ is the set of prime implicates of Σ and G is a clause, then*

$$PS(G,\Sigma) = SUB(\{P - G \mid P \in PI(\Sigma), P \cap G \neq \emptyset \text{ and } P \cup G \text{ is fundamental}\}).$$

is the set of prime supports for G w.r.t. Σ.

Proof : If Σ is inconsistent or G is a tautology, $PS(G, \Sigma) = \emptyset$ and the theorem is true. We assume Σ is consistent and G is fundamental.

Let $S \in PS(G, \Sigma)$. By construction, S is a fundamental support for G. We shall prove that S is minimal. Suppose there is a minimal support S' for G such that $S' \subset S$. By theorem 4.1, $S' = P - G$ for some $P \in MI(\Sigma)$ and $P \cap G \neq \emptyset$. If $S' \cup G$ is fundamental, $P \in PI(\Sigma)$ and $S' \in PS(G, \Sigma)$ contradicting $S \in PS(G, \Sigma)$. If $S' \cup G$ is non-fundamental, there is some literal x such that $\overline{x} \in S'$ and $x \in G$. But then $S' \not\subset S$ since $\overline{x} \notin S$. Therefore, S is minimal.

Conversely, let S be a prime support for G. Since S is also a minimal support for G, by theorem 4.1, there exists a $P \in MI(\Sigma)$ such that $P \cap G \neq \emptyset$ and $S = P - G$. Assume that $P \in MTI(\Sigma)$, then $P = x\overline{x}$ for some $x \in \mathcal{V}$. But then $S \cup G$ is non-fundamental. Therefore $P \in PI(\Sigma)$. Assume $P \cup G$ is non-fundamental. Since P and G are both fundamental, there must exist a literal x such that $x \in P$ and $\overline{x} \in G$. But then $S \cup G$ is non-fundamental. Therefore, $S \in PS(G, \Sigma)$. □

Example 4.2 Let $\Sigma = \{\overline{a}\overline{d}b, \overline{b}\overline{e}c, \overline{p}c\}$ and the set of prime implicates $PI(\Sigma) = \{\overline{a}\overline{d}\overline{e}c, \overline{a}\overline{d}b, \overline{b}\overline{e}c, \overline{p}c\}$. If $G = \overline{d}bc$, then $PS(G,\Sigma) = SUB(\{\overline{a}\overline{e}, \overline{a}, \overline{p}\}) = \{\overline{a}, \overline{p}\}$. Notice that the prime implicate $P = \overline{b}\overline{e}c$ has a non-empty intersection with G but $P \cup G$ is non-fundamental. Also, the fundamental support "$\overline{a}\overline{e}$" is not prime because "$\overline{a}$" subsumes it.

4.2 Trivial Support

As indicated in the previous sections, there are some distinguished types of minimal supports that trivially support a clause G. Intuitively, if $G = a\overline{b}c$, then the negation of its literals "$\overline{a}$", "b" and "$\overline{c}$" are all potentially trivial supports for G. That is, since $\Sigma \models \overline{a} \vee a\overline{b}c$ holds for any Σ, if $\Sigma \not\models \overline{a}$, then "$\overline{a}$" is a trivial support for G.

Definition 4.3 (Trivial Support) *Let Σ be a set of clauses and G a clause. The clause T is a trivial support clause for G if T is a support for G and $T \cup G$ is non-fundamental. The clause T is a minimal trivial support clause for G if T is both a trivial and minimal support for G.*

Recall that according to corollary 2.1 and theorem 2.1, the set $MI(\Sigma)$ is composed of two disjoint sets $PI(\Sigma)$ and $MTI(\Sigma)$. We have already seen $MS(G,\Sigma)$ being constructed from $MI(\Sigma)$, and $PS(G,\Sigma)$ being constructed from $PI(\Sigma)$. Ideally, we would like to have a similar definition for the set of minimal trivial supports for a clause G w.r.t. Σ using $MTI(\Sigma)$ only. Unfortunately, the following observation reveals such impossibility. Let

$$\Delta = SUB(\{M - G \mid M \in MTI(\Sigma) \text{ and } M \cap G \neq \emptyset\}).$$

First, consider $G = x\overline{x}A$ for some $x \in \mathcal{V}$ and fundamental clause A. Since G is a non-funtamental clause and if Σ is consistent, any support for G is a trivial one. Therefore, the only minimal trivial support for G is $\Box$. But, if x or $\overline{x} \in PI(\Sigma)$, then $x\overline{x} \notin MTI(\Sigma)$ and therefore $\Box \notin \Delta$.

The second problem arises when $G = xy$ is a fundamental clause such that $x \in PI(\Sigma)$ and $y \notin PI(\Sigma)$. Obviously, $x\overline{x} \notin MTI(\Sigma)$ and $y\overline{y} \in MTI(\Sigma)$. Since $PS(G,\Sigma) = \{\Box\}$, no trivial support for G is minimal. But, according to Δ, $\overline{y}$ is a minimal trivial support. To overcome these problems we resort to the following less appealing characterization of the set of minimal trivial supports.

Theorem 4.3 *If Σ is a set of clauses and G is a clause, then the set $MTS(G,\Sigma)$ defined as*

1. *if Σ is consistent and G is non-fundamental then $MTS(G,\Sigma) = \{\Box\}$ otherwise*

2. $MTS(G,\Sigma) = \{M - G \mid M \in MTI(\Sigma)$ *,* $M \cap G \neq \emptyset$ *and no clause in* $PS(G,\Sigma)$ *subsumes* $M - G\}$

is the set of minimal trivial supports for G *w.r.t. to* Σ*.*

Proof : If Σ is consistent and G is non-fundamental then the only minimal and trivial support for G is $\Box$, therefore the theorem holds. If Σ is inconsistent, $MTI(\Sigma) = \emptyset$ and the theorem holds. Let Σ be consistent and G be a fundamental clause, and let $S \in MTS(G,\Sigma)$. Since $MTI(\Sigma) \subseteq MI(\Sigma)$, any such S is a support for G (by theorem 4.1) and is trivial by construction (i.e. $S \cup G$ is non-fundamental). Suppose S is not minimal. By construction, S is a single literal x such that $\overline{x} \in G$ and the only clause that subsumes S is $\Box$. But, if $\Box$ is a support for G, $\Box \in PS(G,\Sigma)$ (because G is fundamental) which contradicts the hypothesis $S \in MTS(G,\Sigma)$.

Conversely, let S be a minimal trivial support for G. By definition 4.3 and theorem 4.1, $S \in MS(G,\Sigma)$, i.e. there exists an $M \in MI(\Sigma)$ such that $S = M - G$, $M \cap G \neq \emptyset$ and $S \cup G$ is non-fundamental. Consequently, there exists an $x \in \mathcal{V}$ such that $x \in S \cup G$ and $\overline{x} \in S \cup G$. Assume that $M \in PI(\Sigma)$, since $x, \overline{x}$ cannot both occur in M or in G, therefore either $x \in M$ and $\overline{x} \in G$ or vice versa. Assume that $M = xAB$ and $G = \overline{x}AC$ with $A \neq \emptyset$ and $B \cap C = \emptyset$, hence $S = xB$. Neither x nor $\overline{x}$ can be in $PI(\Sigma)$ since their presence contradicts the assumption that $M \in PI(\Sigma)$. Therefore by theorem 2.2, $M' = x\overline{x} \in MTI(\Sigma)$ which implies that $S' = M' - G = x$ is a support for G and $S' \subset S$ contradicting the assumption that S is a minimal trivial support. Therefore $M \in MTI(\Sigma)$. Moreover, no $P \in PS(G,\Sigma)$ can subsume S because S is minimal. Consequently, $S \in MTS(G,\Sigma)$. $\Box$

Alternatively, the set of minimal trivial supports for a clause G w.r.t. a set Σ can be defined as follows:

1. if Σ is consistent and G is non-fundamental then $MTS(G,\Sigma) = \{\Box\}$
2. if $PS(G,\Sigma) = \{\Box\}$ then $MTS(G,\Sigma) = \emptyset$, otherwise
3. $MTS(G,\Sigma) = \{M - G \mid M \in MTI(\Sigma)$ *and* $M \cap G \neq \emptyset\}$

The following corollary gives an even more simplified way of expressing the set of minimal trivial supports and is used in the algorithm given in the next section.

Corollary 4.2 *If* Σ *is a set of clauses and* G *is a clause, then*

1. if Σ *is consistent and* G *is non-fundamental then* $MTS(G,\Sigma) = \{\Box\}$

2. if $PS(G,\Sigma) = \{\Box\}$ *then* $MTS(G,\Sigma) = \emptyset$*, otherwise*

3. $MTS(G,\Sigma) = \{\overline{x} \mid x \in G \text{ and } \overline{x} \notin PI(\Sigma)\}$

The theorem below explicitly summerizes the relations among the various sets of supports.

Theorem 4.4 *If Σ is a set of clauses and G a clause, then*

$$PS(G,\Sigma) \cap MTS(G,\Sigma) = \emptyset \quad \textit{and} \quad MS(G,\Sigma) = PS(G,\Sigma) \cup MTS(G,\Sigma).$$

Proof : The theorem is a direct consequence of the theorems 2.1, 4.1, 4.2 and 4.3. $\square$

A trivial consequence of the theorem is that if the set $MS(G,\Sigma)$ is available, the sets $PS(G,\Sigma)$, $MTS(G,\Sigma)$ can be computed from

$$MTS(G,\Sigma) = \{T \mid T \in MS(G,\Sigma) \text{ and } T \cup G \text{ is non-fundamental}\}$$

and $PS(G,\Sigma) = \{P \mid P \in MS(G,\Sigma) \; and \; P \cup G \; is \; fundamental\}$.

In summary, we have presented methods to compute the sets $MS(G,\Sigma)$, $PS(G,\Sigma)$ and $MTS(G,\Sigma)$. We have shown that any of these sets can be constructed from the set $PI(\Sigma)$ which justifies it as the only set of implicates of Σ that we need to maintain in order to efficiently compute any set of supports for any clause. Also, the distinction between minimal, trivial and prime supports suggests that under certain circumstances, if the trivial supports for a clause G are of no concern to the *Reasoner* , the *CMS* might as well not compute them at all.

Conversely, if the *Reasoner* has a need for trivial supports for a query G, the *CMS* can compute the set of minimal trivial supports given the set of prime supports. In the case of a single clause query, the minimal trivial support can be easily computed. There is no obvious application for this type of minimal trivial support beside determining the triviality of the query. On the other hand, if the query is a conjunction of clauses, the minimal trivial support for each clause is needed for the construction of minimal supports for the conjunction (section 6). Moreover, observed that the minimal trivial support for a conjunction of clauses is closely related to minimal models (minimal number of truth assignments). For example, if Σ is empty and $\mathcal{F}$ is a CNF formula, then the minimal trivial support S i.e., $\models S \vee \mathcal{F}$ and $\not\models S$ can be shown to correspond to a minimal model of $\mathcal{F}$. We shall leave the investigation into this issue for future research.

5 Query

Recall that a *CMS* query has, for the time being, the form of a single clause G. The *CMS* replies with the set of minimal supports, $MS(G,\Sigma)$, for G with respect to the *CMS* knowledge base Σ. Since trivial and prime supports for G may have different meaning for the *Reasoner* , the *CMS* differentiates among them. To this end, the reply to a query G consists of two disjoint sets: $MTS(G,\Sigma)$ and $PS(G,\Sigma)$. The *Reasoner* can obtain useful information about G by a simple inspection of these sets.

1. If both sets are empty, $\Sigma \cup G$ is inconsistent;
2. if $MTS(G,\Sigma) = \{\Box\}$ and $PS(G,\Sigma) = \emptyset$, G is a tautology;
3. if $PS(G,\Sigma) = \{\Box\}$, G is a logical consequence of Σ
4. if $PS(G,\Sigma) = \emptyset$ and $MTS(G,\Sigma) \neq \emptyset$, G is consistent with Σ but not related to it.

We believe that in most applications only the prime supports are important for the *Reasoner* —the section on diagnostic reasoning supports our claim. Nevertheless, the minimal trivial supports are indispensable for computing the prime supports for conjunctive queries as the later section indicates. The following algorithm is the amalgamation of the results presented thus far.

```
Algorithm for Minimal Supports
Input: a set of clauses $PI(\Sigma)$ and a clause $G$.
Output: $MS(G,\Sigma) = PS(G,\Sigma) \cup MTS(G,\Sigma)$.
Step 1.0 If $PI(\Sigma) = \{\Box\}$ then $MTS(G,\Sigma) = \emptyset$ and $PS(G,\Sigma) = \emptyset$, GOTO 6.0.
Step 2.0 If $G$ is non-fundamental then $PS(G,\Sigma) = \emptyset$ and $MTS(G,\Sigma) = \{\Box\}$, GOTO 6.0.
Step 3.0 $PS(G,\Sigma) = SUB(\{P - G \mid P \in PI(\Sigma), P \cap G \neq \emptyset \textit{ and } P \cup G \textit{ is fundamental}\})$
Step 4.0 If $PS(G,\Sigma) = \{\Box\}$ then $MTS(G,\Sigma) = \emptyset$, GOTO 6.0.
Step 5.0 $MTS(G,\Sigma) = \{\overline{x} \mid x \in G \textit{ and } \overline{x} \notin PI(\Sigma)\}$
Step 6.0 RETURN: $MS(G,\Sigma) = PS(G,\Sigma) \cup MTS(G,\Sigma)$.
```

Example 5.1 We demonstrate the algorithm with the following example. Let $\Sigma = \{\overline{a}\overline{b}cd,$ $\overline{c}fg$, $\overline{d}hj$, $\overline{g}u$, $\overline{h}v$, $w\}$. The set of prime implicates of Σ are

$$
\begin{array}{lllll}
PI(\Sigma) = \{ & (1)\, \overline{a}\overline{b}fujv, & (2)\, \overline{a}\overline{b}fuhj, & (3)\, \overline{a}\overline{b}fgjv, & (4)\, \overline{a}\overline{b}fghj, \\
 & (5)\, \overline{a}\overline{b}cjv, & (6)\, \overline{a}\overline{b}chj, & (7)\, \overline{a}\overline{b}dfu, & (8)\, \overline{a}\overline{b}dfg, \\
 & (9)\, \overline{a}\overline{b}cd, & (10)\, \overline{c}fu, & (11)\, \overline{c}fg, & (12)\, \overline{d}jv, \\
 & (13)\, \overline{d}hj, & (14)\, \overline{g}u, & (15)\, \overline{h}v, & (16)\, w\}.
\end{array}
$$

The following four queries for the set of clauses Σ above illustrate the type of minimal supports obtained using the algorithm. For clarity, each support clause is accompanied by a number indicating which prime implicate in $PI(\Sigma)$ the support is generated from.

Throughout the example, Δ represents the set of potential prime supports for the query i.e., the set constructed at Step 3.0 of the algorithm before the application of the *SUB* operator.

Query A (Minimal Support): $G = \overline{a}jd$

$\Delta = \{\overline{b}fuv(1),\ \overline{b}fuh(2),\ \overline{b}fgv(3),\ \overline{b}fgh(4),\ \overline{b}cv(5),\ \overline{b}ch(6),\ \overline{b}fu(7),\ \overline{b}fg(8),\ \overline{b}c(9)\}$

$PS(G,\Sigma) = \{\overline{b}c(9),\ \overline{b}fg(8),\ \overline{b}fu(7)\}$

$MTS(G,\Sigma) = \{a,\ \overline{j},\ \overline{d}\}$

Notice that in query **A**, the prime implicates (12) and (13) do not contribute to a fundamental support clause because the union of the prime implicate and the query is non-fundamental. In the set $PS(G,\Sigma)$, notice that the prime support clause (7) subsumes (1) and (2); (8) subsumes (3) and (4); and (9) subsumes (5) and (6). The set of minimal trivial supports includes the negations of the goal literals because none of them is in $PI(\Sigma)$. Thus, the set of minimal supports for the clause "$\overline{a}jd$" is the set $PS(G,\Sigma) \cup MTS(G,\Sigma)$.

Query B (Entailment): $G = \overline{h}v$

$\Delta = \{\overline{a}\overline{b}fuj(1),\ \overline{a}\overline{b}fgj(3),\ \overline{a}\overline{b}cj(5),\ \overline{d}j(12),\ \square(15)\}$

$PS(G,\Sigma) = \{\square\}$

$MTS(G,\Sigma) = \emptyset$

In query **B**, the query G is entailed by Σ, that is, by theorem 2.3 there is a prime implicate (15) that subsumes G. Thus the only minimal support for G is the empty clause, as shown in the set Δ and consequently in the set $PS(G,\Sigma)$.

Query C (Tautology): $G = w\overline{w}$

$PS(G,\Sigma) = \emptyset$

$MTS(G,\Sigma) = \{\square\}$

In query **C**, the query is a tautology (non-fundamental). According to the definition of minimal support (definition 4.1), the only minimal support clause for a tautology is the empty clause found in the set $MTS(G,\Sigma)$. This is obtained from Step 2.0 in the algorithm.

Query D (Inconsistent): $G = \overline{w}$

$\Delta = \emptyset$

$PS(G,\Sigma) = \emptyset$

$MTS(G,\Sigma) = \emptyset$

Finally, in case **D**, the query is inconsistent, that is, $\Sigma \cup \overline{G} \models \Box$ or $\Sigma \models \overline{G}$. Such a query does not have any intersection with any prime implicate in $PI(\Sigma)$, consequently $PS(G, \Sigma)$ is the empty set. Also, the negation of G, "w", is the clause (16) in $PI(\Sigma)$ therefore the set $MTS(G, \Sigma)$ is also empty.

5.1 Preference Ordering

Normally, the set of minimal supports for a query contains more than one clause. It would be interesting to design a method to distinguish among these minimal supports, in fact, the notion of preferences in minimal support has been advocated in the literature. For instance in (Cox and Pietrzykowski, 1986), a *basic cause* for a query G is equivalent to our notion of a prime support S for G such that there is no other prime support S' for G that satisfies $\Sigma \models S' \rightarrow S$. In (Poole, 1988), a *least presumptive hypothesis* is comparable to a minimal support S such that there is no other minimal support S' for G that satisfies $\Sigma \models S' \rightarrow S$. Poole (1988) also observed that there is a need for not only basicness and least presumption, but more complete specification of their preference ordering. We shall attemp to provide a preference ordering schema for a set minimal supports for a query w.r.t. Σ.

Recall that subsumption was used as the ordering relation in the set of minimal supports for a query G. Since any support for any clause is by nature fundamental, the same ordering can be used. If S_1 and S_2 are supports for G w.r.t. Σ, S_1 subsumes S_2 if $\models S_1 \rightarrow S_2$. Note that this ordering relation is independent of any set Σ. Interestingly enough, we can further define a more restricted ordering relation among the elements of $MS(G, \Sigma)$ that depends on Σ as follows.

Definition 5.1 (Minimal Support Preference Ordering) *Let Σ be a set of clauses, G a clause, $MS(G, \Sigma)$ the set of all minimal supports for G w.r.t. Σ and $\Phi \subseteq \Sigma$. We say that*

1. *S_1 precedes S_2 w.r.t. Φ ($S_1 \leadsto S_2$) if $\Phi \models S_1 \rightarrow S_2$ for distinct $S_1, S_2 \in MS(G, \Sigma)$.*
2. *S is an upper minimal support w.r.t. Φ ($S \leadsto$) if there is no other $S' \in MS(G, \Sigma)$ such that $\Phi \models S \rightarrow S'$.*
3. *S is a lower minimal support w.r.t. Φ ($\leadsto S$) if there is no other $S' \in MS(G, \Sigma)$ such that $\Phi \models S' \rightarrow S$.*

If S is both upper ($S \leadsto$) and lower ($\leadsto S$) minimal support w.r.t. Φ, then S is called isolated and is denoted by itself. Also, if $\Phi = \Sigma$, the resulting ordering is called a canonical ordering of $MS(G, \Sigma)$.

The above definition allows the *Reasoner* to provide a set of ordering constraints (depending on the application domain) which together with Φ allows some degree of discrimination among the minimal supports. In the absence of such constraints, Σ can always be used to serve the purpose. From here on, we will concentrate on canonical orderings and note that theorem 2.3 ensures that for any $A, B \in MS(G, \Sigma)$, $A \rightsquigarrow B$ iff there exists a $P \in PI(\Sigma)$ that subsumes $\overline{A} \vee B$.

Example 5.2 Consider Query **A** in example 5.1. $G = \overline{a}jd$ and $MS(G, \Sigma) = \{\overline{b}c, \overline{b}fg, \overline{b}fu, a, \overline{j}, \overline{d}\}$. The canonical ordering forces the following relations among the elements of $MS(G, \Sigma)$:

$$\begin{array}{lllll} \overline{b}c \rightsquigarrow \overline{b}fg\ , & \overline{b}c \rightsquigarrow \overline{b}fu\ , & \overline{b}fg \rightsquigarrow \overline{b}fu\ , & \overline{b}fu \rightsquigarrow\ , & \rightsquigarrow \overline{b}c\ , \\ a \quad , & \overline{j} \quad , & \overline{d} & & \end{array}$$

The graph in figure 3 provides a visual presentation of the preference ordering on $MS(G, \Sigma)$. The arrows in the graph correspond directly to the preference ordering relation $\rightsquigarrow$. The minimal trivial supports "a", "$\overline{j}$" and "$\overline{d}$" are isolated; the prime support "$\overline{b}c$" is the lower and "$\overline{b}fu$" is the upper minimal support w.r.t. to Σ.

$$\begin{tikzcd}
& \overline{b}c \arrow[dl] \arrow[dr] & \\
\overline{b}fg \arrow[rr] & & \overline{b}fu \\
a & \overline{j} & \overline{d}
\end{tikzcd}$$

Figure 3: Canonical Ordering in $MS(G, \Sigma)$

We have mentioned earlier that in many applications only the set of prime supports is of interest, consequently, a preference ordering for the prime supports is needed instead. By restricting the ordering relation defined in this section (definition 5.1) to the set $PS(G, \Sigma)$, a similar preference ordering on the set of the prime supports for a query G can be obtained.

6 Conjunctive Queries

To generalize the functionality of the *CMS* , Reiter and de Kleer (pp 185, 1987) proposed the notion of a minimal support for a conjunction of clauses. In this section, we shall present a method for computing the proposed set of minimal supports. Initially, we shall present the method for finding the minimal support for a conjunction of two clauses and subsequently, generalize it to multiple conjuncts. The notion of a support for a conjunction of two clauses $G_1 \wedge G_2$ is derived from the definition of support (definition 4.1) by substituting $G = G_1 \wedge G_2$. Intuitively, the set of support for $G_1 \wedge G_2$ can be constructed by first constructing the set of minimal supports $MS(G_1, \Sigma)$ and $MS(G_2, \Sigma)$ respectively, and taking the pairwise union (disjunction) of their minimal supports provided they satisfy certain conditions. In the following, if A and B are clauses, $A \vee B$ denotes the clause obtained by joining the literals of A and B without repetitions.

Theorem 6.1 *Let Σ be a set of clauses and $G = G_1 \wedge G_2$ the conjunction of two clauses. Let*

$$\Delta = \{S_1 \vee S_2 \mid S_1 \in MS(G_1, \Sigma)\ ,\ S_2 \in MS(G_2, \Sigma) \textit{ and no } M \in MI(\Sigma) \textit{ subsumes } S_1 \vee S_2\}.$$

The set $MS(G, \Sigma) = SUB(\Delta)$ is the set of minimal supports for G w.r.t. to Σ.

Proof : Let $S \in MS(G, \Sigma)$. We shall prove that S is a minimal support clause for G.

a. (support) By definition, $S = S_1 \vee S_2$ where $S_1 \in MS(G_1, \Sigma)$ and $S_2 \in MS(G_2, \Sigma)$. Since $\Sigma \models S_1 \vee G_1$ and $\Sigma \models S_2 \vee G_2$, then by propositional reasoning $\Sigma \models (S_1 \vee S_2) \vee (G_1 \wedge G_2)$[5]. By the construction of $MS(G, \Sigma)$, there is no other minimal implicate that subsumes S and hence, by theorem 2.3, $\Sigma \not\models S$. Consequently, S is a support for G w.r.t. Σ.

b. (minimality) Let R be a support for G, i.e. $\Sigma \models R \vee (G_1 \wedge G_2)$ and $\Sigma \not\models R$, such that $R \subset S$. Since $\Sigma \models R \vee G_1$ and $\Sigma \models R \vee G_2$, R is a support for G_1 and G_2 respectively. Hence there exist a $R_1 \in MS(G_1, \Sigma)$ and a $R_2 \in MS(G_2, \Sigma)$ such that $R_1 \subseteq R$ and $R_2 \subseteq R$. Therefore $R_1 \vee R_2 \subseteq R$ and since $\Sigma \not\models R$, by theorem 2.3 there is no $M \in MI(\Sigma)$ that subsumes $R_1 \vee R_2$. This means that $R_1 \vee R_2 \in \Delta$. Therefore either $R_1 \vee R_2 \in MS(G, \Sigma)$ or it is subsumed by some $R' \in MS(G, \Sigma)$. Since $R' \subseteq R_1 \vee R_2 \subset S$, S is subsumed by some element in $MS(G, \Sigma)$ contradicting the assumption that $S \in MS(G, \Sigma)$.

Conversely, let S be a minimal support for $G = G_1 \wedge G_2$ w.r.t. Σ. We shall prove that $S \in MS(G, \Sigma)$. Since $\Sigma \models S \vee (G_1 \wedge G_2)$, hence $\Sigma \models S \vee G_1$, $\Sigma \models S \vee G_2$ and $\Sigma \not\models S$ i.e., S is a support for G_1 and G_2 respectively. By the definition of minimal support (definition 4.1), there exist $S_1 \subseteq S$ and $S_2 \subseteq S$ such that S_1, S_2 are minimal supports for G_1, G_2 respectively i.e. $S_1 \in MS(G_1, \Sigma)$ and $S_2 \in MS(G_2, \Sigma)$. Since $\Sigma \models S_1 \vee S_2 \vee G$ and $\Sigma \not\models S_1 \vee S_2$ because $S_1 \cup S_2 \subseteq S$, no $M \in MI(\Sigma)$ subsumes $S_1 \vee S_2$, hence $S_1 \vee S_2$ is a

[5] $\models (\overline{S_1} \to G_1) \wedge (\overline{S_2} \to G_2) \to ((\overline{S_1} \wedge \overline{S_2}) \to (G_1 \wedge G_2))$.

support for G and also $S_1 \vee S_2 \in \Delta$. If $S_1 \vee S_2 \subset S$, then S is not minimal contradicting the assumption that S is, consequently, $S = S_1 \vee S_2$ and therefore $S \in MS(G, \Sigma)$. $\Box$.

Example 6.1 Let $\Sigma = \{\overline{p}q, \overline{q}r, \overline{p}t\}$ and $G = r \wedge t$. Then

$$\begin{aligned}
&PI(\Sigma) = \{\overline{p}q, \overline{q}r, \overline{p}t, \overline{p}r\} \\
&MS(r, \Sigma) = PS(r, \Sigma) \cup MTS(r, \Sigma) = \{\overline{q}, \overline{p}\} \cup \{\overline{r}\} \\
&MS(t, \Sigma) = PS(t, \Sigma) \cup MTS(t, \Sigma) = \{\overline{p}\} \cup \{\overline{t}\} \\
&\Delta = \{\overline{q}\overline{p}, \overline{q}\overline{t}, \overline{p}, \overline{p}\overline{t}, \overline{r}\overline{p}, \overline{r}\overline{t}\} \\
&MS(r \wedge t, \Sigma) = \{\overline{q}\overline{t}, \overline{p}, \overline{r}\overline{t}\}
\end{aligned}$$

An obvious generalization of theorem 6.1 can be obtained as follows. If $G_1, \ldots, G_n$ are n clauses and $G = G_1 \wedge \ldots \wedge G_n$, then we define

$$\begin{aligned}
MS(G, \Sigma) = SUB(\{\ & S_1 \vee S_2 \vee \ldots \vee S_n \mid \\
& \text{for each } i,\ 1 \leq i \leq n,\ S_i \in MS(G_i, \Sigma) \text{ and} \\
& \text{no } M \in MI(\Sigma) \text{ subsumes } S_1 \vee S_2 \vee \ldots \vee S_n\}).
\end{aligned}$$

Notice that in the above method, if $S_1 \vee S_2$ is subsumed by some $S_1' \vee S_2'$ where $S_1, S_1' \in MS(G_1, \Sigma)$, $S_2, S_2' \in MS(G_2, \Sigma)$ and $\Psi = \bigvee_{i=3}^{n} S_i \in MS(G_i, \Sigma)$, then obviously

$$S_1' \vee S_2' \vee \Psi \ \textit{subsumes}\ S_1 \vee S_2 \vee \Psi$$

which means that we have generated a lot of non-minimal supports that are subsumed later. The above scenario suggests that we should remove non-minimal supports as early as possible preventing the unnecessary combinatorial explosion.

The following theorem is a generalization of theorem 6.1 and gives the basis for a recursive algorithm for computing the minimal supports for the conjunction of an arbitrary (finite) number of clauses. The theorem exploits the local non-minimality condition and remove non-minimal supports as soon as possible.

Theorem 6.2 *If Σ is a set of clauses and $G_1, \ldots, G_n, n \geq 2$ are n clauses, the set of minimal supports for $G = G_1 \wedge \ldots \wedge G_n$ can be defined as*

$$\begin{aligned}
MS(G_1 \wedge \ldots \wedge G_n, \Sigma) = SUB(\{S \vee S' \mid\ & S \in MS(G_1 \wedge \ldots \wedge G_{n-1}, \Sigma) \textit{ and} \\
& S' \in MS(G_n, \Sigma) \textit{ and} \\
& \textit{no } M \in MI(\Sigma) \textit{ subsumes } S \vee S'\}).
\end{aligned}$$

Proof : By simple induction on the number of clauses in the conjunction and theorem 6.1. □

Notice that the condition "no $M \in MI(\Sigma)$ subsumes $S \vee S'$" can actually be replaced by (i) $S \vee S'$ is fundamental and (ii) no $P \in PI(\Sigma)$ subsumes $S \vee S'$. The above replacement is an obvious consequence of theorem 2.3 and the following corollary restates theorem 6.2 in a simplified manner.

Corollary 6.1 *If Σ is a set of clauses and $G_1, \ldots, G_n, n \geq 2$ are n clauses, the set of minimal supports for $G = G_1 \wedge \ldots \wedge G_n$ can be defined as*

$$\begin{aligned} MS(G_1 \wedge \ldots G_n, \Sigma) = SUB(\{S \vee S' \mid\ & S \in MS(G_1 \wedge \ldots \wedge G_{n-1}, \Sigma) \textit{ and} \\ & S' \in MS(G_n, \Sigma) \textit{ and} \\ & \textit{no } P \in PI(\Sigma) \textit{ subsumes } S \vee S' \textit{ and} \\ & S \vee S' \textit{ is fundamental } \}). \end{aligned}$$

Example 6.2 In example 6.1, if the query is $G = r \wedge t \wedge q$, we can construct the minimal support for G by taking the set of minimal supports $MS(r \wedge t, \Sigma)$ computed in example 6.1, and pairing with the set $MS(q, \Sigma)$ as follows:

$$\begin{aligned} & MS(r \wedge t, \Sigma) = \{\overline{q}\overline{t},\ \overline{p},\ \overline{r}\overline{t}\} \\ & MS(q, \Sigma) = \{\overline{p},\ \overline{q}\} \\ & \Delta = \{\overline{p}\overline{q}\overline{t},\ \overline{q}\overline{t},\ \overline{p},\ \overline{p}\overline{q},\ \overline{p}\overline{r}\overline{t},\ \overline{q}\overline{r}\overline{t}\} \\ & MS(r \wedge t \wedge q, \Sigma) = \{\overline{q}\overline{t},\ \overline{p}\} \end{aligned}$$

Before we leave this section we present an alternative for computing the set of minimal supports for a finite conjunction of clauses[6].

Theorem 6.3 *Let Σ be a set of clauses and $G = G_1 \wedge \ldots \wedge G_n$ a conjunction of $n \geq 2$ clauses. For any clause S, $S \in MS(G, \Sigma)$ iff $S \in PS(\sigma, \Sigma')$ where, σ is a new propositional variable not occuring in $\Sigma \cup G$ and $\Sigma' = \Sigma \cup \{G \rightarrow \sigma\}$.*

Proof : We need to prove that for any clause S which contains no occurences of σ, S is a support for σ w.r.t. Σ' iff S is a support for G w.r.t. Σ.

Assume S is a support for σ w.r.t. Σ', then $\Sigma' \models S \vee \sigma$ or $\Sigma \cup \{G \rightarrow \sigma\} \models S \vee \sigma$ and $\Sigma \models (G \rightarrow \sigma) \rightarrow S \vee \sigma$. By propositional reasoning, $\Sigma \models S \vee G \vee \sigma$, since σ does not occur

[6]This alternative was brought to our attention by Prof. Raymond Reiter in a personal communication on October 1988.

in either of Σ, S or G, $\Sigma \models S \vee G$. Moreover, since $\Sigma' \not\models S$ and $\Sigma \not\models S$, therefore S is a support for G w.r.t. Σ.

Conversely, assume S is a support for G w.r.t. Σ, i.e. $\Sigma \models S \vee G$ and $\Sigma \not\models S$, then $\Sigma' \models (S \vee G) \wedge (G \rightarrow \sigma)$ and $\Sigma' \models S \vee \sigma$. In addition $\Sigma' \not\models S$, for otherwise $\Sigma \models (G \rightarrow \sigma) \rightarrow S$ which by propositional reasoning implies that $\Sigma \models S \vee \overline{\sigma}$. Since σ does not occur in Σ or S, $\Sigma \models S$ which contradicts our hypothesis. Consequently S is a support for σ w.r.t. Σ'. □

The recursive algorithm implied by theorem 6.2, although it looks complicated, it is actually better in complexity terms than the method proposed by theorem 6.3. The latter technique requires first transforming the (DNF) formula $G \rightarrow \sigma$ into an equivalent formula $\mathcal{F}$ in CNF, and then computing the prime implicates of $\Sigma \cup \mathcal{F}$. This transformation has a combinatorial explosion in general. Moreover, if $\mathcal{F}$ has m clauses, m applications of the incremental algorithm are required to obtain the new set of prime implicates. This also implies that for each query, the knowledge base Σ must be duplicated before adding the augmented query clause in order to preserve the set Σ for other queries.

The algorithms discussed in this section generate the set of minimal supports for a conjunction of clauses in a rather direct way. Unlike the algorithm in section 5, these algorithms do not explicitly generate the sets of prime and minimal trivial supports for a conjunction, therefore, some extra effort is needed to partition the minimal supports into prime and trivial ones. More specifically, let Σ be a set of clauses and $G = G_1 \wedge \ldots \wedge G_n$ is a conjunction of $n \geq 2$ clauses. By theorem 4.4, for any clause $S \in MS(G, \Sigma)$, $S \in PS(G, \Sigma)$ iff for some i, $1 \leq i \leq n$, $G_i \cup S$ is fundamental, otherwise $S \in MTS(G, \Sigma)$. Consequently, when a conjunctive query G is presented to the *CMS* , it computes $MS(G, \Sigma)$, applies the fundamentality test on $MS(G, \Sigma)$ and replies with the sets $PS(G, \Sigma)$ and $MTS(G, \Sigma)$.

7 Diagnostic Reasoning

In this section, we shall demonstrate the utilization of the *CMS* in diagnostic reasoning. Diagnosis is an act of investigation or analysis of the cause or nature of a condition, situation, or problem. A diagnostic reasoning system as studied in Artificial Intelligence is a computational system that performs diagnosis. In rectrospect, conventional diagnostic systems employed some form of rule-based production system. The knowledge encoded in such system is usually in the form of a relation between possible *causes* and possible *effects*.[7] Many new strategies have emerged in the realm of diagnostic reasoning (Milne, 1987);

[7]Usually, *causes* represent the possible states of the system components (malfunction, correct states etc.) or possible diseases in the medical domain, while *effects* are the results of the *causes*.

nevertheless we shall concentrate on the use of the *CMS* in two logic based methodologies: consistency based and explanation based diagnostic reasoning. These paradigms are chosen for the purpose of exemplifying the functionality of *CMS* . A comprehensive study on the adequacy of these paradigms can be found in (Poole 1988a).

To simplify the discussion, we assume a propositional diagnostic *Reasoner* and leave the issues of the protocol between the 1st order *Reasoner* and the *CMS* as future research. The system's knowledge or system description) is encoded as a set of formulae (SD) expressing the ontology of the domain and its tasks. The strategy of the encoding can be (i) *causes* $\rightarrow$ *effects* or (ii) *effects* $\rightarrow$ *causes* where *causes* and *effects* are formulae. Without lost of generality, we will assume a *cause* to be a literal. We shall designate a set of *causes* $\mathcal{C} = \{cause_1, \ldots, cause_n\}$ where each $cause_i$ is a disguintished literal. Furthermore, an observation (OBS) is a set of formulae expressing the observed behavior of the system and of course is related to *effects*. Occassionally, there is also a set of constraints C embedded in SD expressing the relation between *causes*, for example, certain *causes* are mutually exclusive w.r.t. the same *effect*.

7.1 Consistency Based Diagnostic Reasoning

In the consistency based diagnostic reasoning paradigm (Reiter, 1987), the system to be diagnosed is described by formulae of the form *effects* $\rightarrow$ *causes* [8]. The objective of the diagnosis is to extract every minimal subset of *causes* that is consistent with the system description SD and the observation OBS. More formally, a *diagnosis* for (SD, OBS) is a minimal set of *causes* $\Delta \subseteq \mathcal{C}$ such that

$$SD \bigcup \{cause \mid cause \in \Delta\} \bigcup \{\neg cause \mid cause \in \mathcal{C} - \Delta\} \bigcup OBS \qquad (1)$$

is consistent.

A set $T \subseteq \mathcal{C}$ is an *inconsistent subset of causes* with respect to SD and OBS if $SD \cup OBS \cup \{\neg cause \mid cause \in T\}$ is inconsistent. Such a set T is a *minimal inconsistent subset of causes* if no proper subset of it is an inconsistent subset of causes[9]. Thus if $SD \cup OBS \cup \{\neg cause_1, \ldots, \neg cause_k\}$ is inconsistent or $SD \cup OBS \models cause_1 \vee \ldots \vee cause_k$, then using theorem 2.3 and 2.4, there is a prime implicate P of $PI(SD \cup OBS)$ that

[8]Some authors claim that such a representation describes the normal state of the system. The reason for this can be illustrated by an example. If *arthiritis* is known to cause *aching_elbow*, the formula *aching_elbow* $\rightarrow$ *arthiritis* is equivalent to $\neg$*arthiritis* $\rightarrow$ $\neg$*aching_elbow*, which describes the condition of a normal person w.r.t. *arthiritis*.

[9]de Kleer (1986) refered to it as nogoods and Reiter (1987) refered to it as minimal conflict set.

subsumes $cause_1 \vee \ldots \vee cause_k$. Also, by the minimality of prime implicate, we know that the causes in P are a minimal inconsistent subset of causes.

There is a close relationship between the minimal inconsistent subsets of causes and the set of diagnoses for (SD, OBS). Notice that in equation (1), the set of causes $\mathcal{C} - \Delta$ cannot be a superset of a minimal inconsistent subset T for otherwise equation (1) is inconsistent. Assuming we have all the minimal inconsistent subsets of causes, say T_1, T_2 and T_3 as shown in figure 4, then equation (1) is consistent if the set $\mathcal{C} - \Delta$ is not a superset of any minimal inconsistent subset T_i, that is, for every T_i, $T_i \cap \Delta \neq \emptyset$.

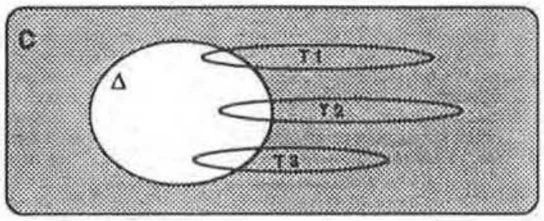


Figure 4: Computing Diagnoses

Suppose $T_1, \ldots, T_n$ are all the minimal inconsistent subsets of causes w.r.t. SD and OBS, where $T_i = \{cause_{i1}, \ldots, cause_{ik_i}\}$ for $1 \leq i \leq n$. Let $T_i' = cause_{i1} \vee \cdots \vee cause_{ik_i}$ for $1 \leq i \leq n$. Clearly, $SD \cup OBS \models T_1' \wedge \cdots \wedge T_n'$. Finally, let $D = D_1 \vee \cdots \vee D_m$ be a minimal (number of conjuncts) DNF formula equivalent to $T_1' \wedge \cdots \wedge T_n'$. Since there are not complementary literals among the T_i''s and consequently among the D_j, D is unique and minimal and can be computed by the technique of normal form transformation and subsumption[10]. It can be shown, following Reiter's (1987) theorem 4.4, the set $\{D_1, \ldots, D_m\}$ is the set of diagnoses for (SD, OBS).

In this framework the *CMS* can be used to compute all the minimal inconsistent subsets of causes as follows. When an observation OBS is made, the ***Reasoner*** transmits to the *CMS* all the clauses that are *related* to OBS. A clause is *related* to OBS in case it contains non-logical symbols (propositional symbols) that occur in OBS or occur in another clause that is related to OBS[11].

Obviously, the observation OBS is also transmitted to the *CMS* and in addition, the

[10]The existence of complementary literals in D does not ensure minimality and therefore uniqueness. This problem is similar to the minimization of Boolean functions in switching theory (Bartee *et al.*, 1962).

[11]Craig's interpolation lemma (Shoenfield, pp 80, 1967) can be used to show that these clauses are sufficient for our purpose. Note that the ***Reasoner*-*CMS*** protocol we describe here is a rather simple and therefore inefficient one. A thorough study of the ***Reasoner*-*CMS*** protocols is among the issues for our future research.

Reasoner supplies the *CMS* with a set $\mathcal{C}$ of distinguished literals that represent *causes*. The *CMS* computes the set of prime implicates of its database and returns a subset of prime implicates such that each prime implicate P consists solely of *causes* ($P \subseteq \mathcal{C}$) to the *Reasoner*. As discussed earlier, these prime implicates are the minimal inconsistent subsets of causes. The *Reasoner* then computes diagnoses from the set of minimal inconsistent subsets of causes using the normal form transformation technique.

Clearly, any fast transformation method from CNF to DNF is suitable for this purpose. For example, we can represent the CNF formula as a matrix and use the connection method (Bibel, 1987) to construct a set of all paths $\mathcal{P}$ through the matrix. It can be shown easily that the minimal DNF formula is the set of all paths $\mathcal{P}'$ that are not subsumed by other paths in the set $\mathcal{P}$. Furthermore, such a technique can be optimized for our setting since no complimentary literals exist in clauses, and subsumptions can be greatly reduced by examining the structure of the matrix.

Finally, observations (or measurements) in incremental fashion can also be accommodated in our framework. A measurement is simply treated as an additional observation. Whenever a measurement M is performed, the result is transmitted to the *CMS* which incrementally computes the new set of prime implicates and subsequently, minimal inconsistent subsets of *causes*, which in turn may lead to a new set of diagnoses.

Example 7.1 A medical diagnosis involves identifying a disease or illness from its signs and symptoms. The following set of propositional formulae, taken and simplified from (Poole *et al.*, 1986), describes the type of symptoms (*effects*) that are produced by several diseases (*causes*).

$$
\begin{aligned}
&\mathcal{C} = \{tennis\_elbow,\ dishpan\_hands,\ arthritis\}.\\
&SD = \{\ aching\_elbow \rightarrow tennis\_elbow \vee arthritis,\\
&\qquad aching\_hands \rightarrow dishpan\_hands \vee arthritis,\\
&\qquad aching\_knee \rightarrow arthritis\}.\\
&OBS = \{\ aching\_elbow,\ aching\_hands\}.\\
&PI(SD \cup OBS) = \{\ tennis\_elbow \vee arthritis, \qquad (\star)\\
&\qquad\qquad dishpan\_hands \vee arthritis, \qquad (\star)\\
&\qquad\qquad aching\_hands,\ aching\_elbow,\ aching\_knee \vee arthritis\}.
\end{aligned}
$$

The prime implicate indicated by ($\star$) is a clause consists solely of *causes* of in $\mathcal{C}$ (the minimal inconsistent subset of causes). The conjunction of the minimal inconsistent subset of *causes* is $T = (tennis\_elbow \vee arthritis) \wedge (dishpan\_hands \vee arthritis)$ and the minimal DNF of T is $D = arthritis \vee (tennis\_elbow \wedge dishpan\_hands)$ which gives two minimal *causes* for the observation namely, "*arthritis*" alone or "*tennis_elbow* $\wedge$ *dishpan_hands*".

Example 7.2 A second example deals with circuit diagnosis. Consider a full adder (Reiter, 1987) shown in figure 5. The gates X_1 and X_2 are *xor* gates; A_1 and A_2 are *and* gates; and O_1 is an *or* gate.

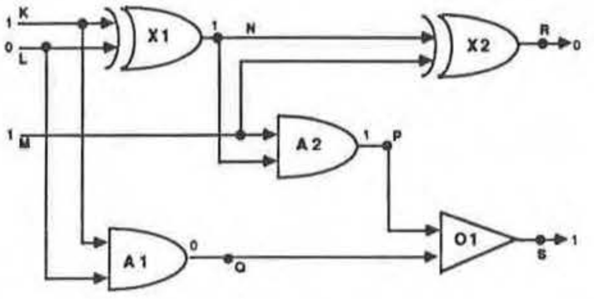


Figure 5: A Full Adder

The system is described be the following set of clauses:

$C = \{ab(X_1),\ ab(X_2),\ ab(A_1),\ ab(A_2),\ ab(O_1)\}$
$SD = \{$

$\neg(K = 0) \vee \neg(L = 0) \vee ab(X_1) \vee (N = 0),$
$\neg(K = 0) \vee (L = 0) \vee ab(X_1) \vee \neg(N = 0),$
$(K = 0) \vee \neg(L = 0) \vee ab(X_1) \vee \neg(N = 0),$
$(K = 0) \vee (L = 0) \vee ab(X_1) \vee (N = 0),$

$\neg(N = 0) \vee \neg(M = 0) \vee ab(X_2) \vee (R = 0),$
$\neg(N = 0) \vee (M = 0) \vee ab(X_2) \vee \neg(R = 0),$
$(N = 0) \vee \neg(M = 0) \vee ab(X_2) \vee \neg(R = 0),$
$(N = 0) \vee (M = 0) \vee ab(X_2) \vee (R = 0),$

$(K = 0) \vee (L = 0) \vee ab(A_1) \vee \neg(Q = 0),$
$\neg(K = 0) \vee ab(A_1) \vee (Q = 0),$
$\neg(L = 0) \vee ab(A_1) \vee (Q = 0),$

$(M = 0) \vee (N = 0) \vee ab(A_2) \vee \neg(P = 0),$
$\neg(M = 0) \vee ab(A_2) \vee (P = 0),$
$\neg(N = 0) \vee ab(A_2) \vee (P = 0),$

$\neg(P = 0) \vee \neg(Q = 0) \vee ab(O_1) \vee (S = 0),$
$(P = 0) \vee ab(O_1) \vee \neg(S = 0),$
$(Q = 0) \vee ab(O_1) \vee \neg(S = 0)\ \}$

The *causes* considered here are possible abnormalities of the components and are represented by the predicate $ab(\_)$. The system description contains, for each component, a set of clauses that describe the normal state (correct function) of the component. Although not shown explicitly, every clause in SD is in the form *effects* $\rightarrow$ *causes* . As an illustration, the first clause in SD describing gate X_1 is equivalent to "$(K = 0) \wedge (L = 0) \wedge (N = 1) \rightarrow ab(X_1)$". Similarly, the second clause for gate X_1 is the same as "$(K = 0) \wedge (L = 1) \wedge (N = 0) \rightarrow ab(X_1)$". For simplicity, we have chosen to use a one value system i.e., for every wire in the sytem it can be 0 or not equal to 0, thus the fact that a wire A has a value 1 is represented by the proposition $\neg(A = 0)$. Alternatively one can use 0 and 1 value together with some additional clauses expressing the constraint that each wire has at most one value.

When the observation[12] $OBS = \neg(K = 0) \wedge (L = 0) \wedge \neg(M = 0) \wedge \neg(R = 0) \wedge (S = 0)$ is transmitted to the *CMS*, it computes the the set of prime implicates

$$\begin{array}{l} PI(SD \cup OBS) = \{ \\ \quad \begin{array}{ll} ab(A_2) \vee \neg(P = 0) \vee ab(X_1), & ab(A_2) \vee ab(X_1) \vee ab(O_1), \quad (\star) \\ ab(X_2) \vee ab(A_2) \vee (P = 0), & ab(X_1) \vee \neg(N = 0), \\ ab(X_2) \vee ab(X_1), \quad (\star) & ab(A_1) \vee (Q = 0), \\ \neg(N = 0) \vee ab(A_2) \vee (P = 0), & (N = 0) \vee ab(A_2) \vee \neg(P = 0), \\ (N = 0) \vee ab(A_2) \vee ab(O_1), & (N = 0) \vee ab(X_2), \\ (Q = 0) \vee ab(O_1), & (P = 0) \vee ab(O_1), \\ \multicolumn{2}{l}{\neg(K = 0), \ \neg(R = 0), \ \neg(M = 0), \ L = 0, \ S = 0\}} \end{array} \end{array}$$

and returns the prime implicates "$ab(A_2) \vee ab(X_1) \vee ab(O_1)$" and "$ab(X_2) \vee ab(X_1)$" which constitute the minimal inconsistent subsets of causes. Consequently, by transfromation we obtain "$ab(A_2) \wedge ab(X_2)$", "$ab(O_1) \wedge ab(X_2)$" and "$ab(X_1)$" as the diagnoses for (SD, OBS).

Suppose now that point P was measured and found to have value 1. Let $\Pi = PI(SD \cup OBS)$, the clause $\neg(P = 0)$ is sent to the *CMS* and the *CMS* incrementally generates the new set of prime implicates

$$\begin{array}{l} PI(\Pi \cup \{\neg(P = 0)\}) = \{ \\ \quad \begin{array}{lll} ab(X_1) \vee \neg(N = 0), & \neg(N = 0) \vee ab(A_2), & ab(A_1) \vee (Q = 0), \\ (N = 0) \vee ab(X_2), & ab(X_2) \vee ab(X_1), \quad (\star) & ab(X_2) \vee ab(A_2), \quad (\star) \\ \multicolumn{3}{l}{\neg(M = 0), \ \neg(K = 0), \ \neg(R = 0), \ \neg(P = 0), \ S = 0, \ L = 0, \ ab(O_1) \quad (\star)\}.} \end{array} \end{array}$$

The prime implicates "$ab(X_2) \vee ab(X_1)$", "$ab(X_2) \vee ab(A_2)$" and "$ab(O_1)$" are the new minimal inconsistent subsets of *causes* making "$ab(X_2) \wedge ab(O_1)$" and "$ab(X_1) \wedge ab(A_2) \wedge ab(O_1)$" the new diagnoses for the added observation.

7.2 Explanation Based Diagnostic Reasoning

In explanation based diagnostic reasoning paradigm (Poole, 1988b), the system SD is described by formulae of the form *causes* $\rightarrow$ *effects* together with some possible constraints. Given an observation OBS, a diagnosis for (SD, OBS) is a minimal conjunction of *causes* E such that $SD \models E \rightarrow OBS$ and $SD \cup E$ is consistent. Such a conjunct E is called a *minimal explanation for OBS with respect to SD*. Obviously, a conjunct E is a minimal explanation for OBS w.r.t. SD iff $\overline{E}$ (the negation of E) is a minimal support for OBS w.r.t. SD, that contains only the negations of causes. We shall call these supports *cause*

[12]Note that for the consistency method, the observation has to be represented as the conjunction "*input* $\wedge$ *output*" instead of the more natural "*input* $\rightarrow$ *output*".

based minimal support for OBS (Tsiknis & Kean, 1989). Indeed, $SD \models E \rightarrow OBS$ iff $SD \models \overline{E} \vee OBS$, and $SD \cup E$ is consistent iff $SD \not\models \overline{E}$ given that SD alone is consistent.

The role of the *CMS* in this framework is now clear. The *Reasoner* transmits to the *CMS* the clauses in SD that are related to the observation and indicates to the *CMS* the set of distinguished literals that denote *causes.* When the *Reasoner* requests for the cause based minimal supports of the observation OBS, the *CMS* computes them by the method described earlier for computing minimal supports with the following restriction: consider only the minimal support $S = P - OBS$ that consists solely of the negations of *causes* $(\overline{S} \subseteq \mathcal{C})$[13]. Thus, the *Reasoner* negates each cause based minimal support and obtains all the diagnoses for (SD, OBS). Similarly, measurements are treated as additional observations. More specifically, if a measurement M is performed, the new diagnoses are obtained as the minimal explanations for $OBS \wedge M$ w.r.t. SD.

Example 7.3 The domain of example 7.1 has the following description in this paradigm.

$$\begin{aligned}
\mathcal{C} &= \{tennis\_elbow,\ dishpan\_hands,\ arthritis\}.\\
SD &= \{\ tennis\_elbow \rightarrow aching\_elbow,\quad dishpan\_hands \rightarrow aching\_hands,\\
&\qquad arthritis \rightarrow aching\_elbow,\quad arthritis \rightarrow aching\_hands,\\
&\qquad arthritis \rightarrow aching\_knee\}.
\end{aligned}$$

Suppose we observe $OBS = aching\_elbow \wedge aching\_hands$. Since

$$\begin{aligned}
PI(SD) = \{\ &\neg dishpan\_hands \vee aching\_hands,\quad \neg arthritis \vee aching\_hands,\\
&\neg tennis\_elbow \vee aching\_elbow,\quad \neg arthritis \vee aching\_elbow,\\
&\neg arthritis \vee aching\_knee\},
\end{aligned}$$

the set $\{\ \neg arthritis,\ \ \neg tennis\_elbow \vee \neg dishpan\_hands\ \}$ is the set of cause based minimal supports for the observation. Thus, by transformation, we obtain "$arthritis$" and "$tennis\_elbow \wedge dishpan\_hands$" as the diagnoses for (SD, OBS).

Example 7.4 We now consider the system presented in example 7.2. The complete system description as well as the observation are shown in the appendix. Note that in this paradigm, SD describes both the normal and erroneous states. Specifically, for each component in the circuit, SD contains a set of clauses that define the normal state of the component (as in example 7.2) followed by a similar set that specifies the faulty state of that component. Additionally, the observation is in the form *input* $\rightarrow$ *output* which is a more natural description of the state of a circuit component. The interested reader can find discussions on these issues in (Poole, 1988a) and note that the representation presented above is crucial for the correctness of this paradigm.

[13]Alternatively, we could use the *CMS* to generate all the minimal supports for OBS and assign to the *Reasoner* the responsibility of filtering out the cause based ones.

In this example, the procedure for computing the diagnoses deviates slightly from the general procedure set out in this section. For efficiency reasons, instead of computing the minimal explanations of OBS w.r.t. SD, we compute the minimal explanations of *output* w.r.t. $SD \cup \{input\}$. This is justified by the fact that for any E, $SD \models E \rightarrow (input \rightarrow output)$ iff $SD \models (E \wedge input) \rightarrow output$ or simply $SD \cup \{input\} \models E \rightarrow output$. Consequently, when

$$OBS = \neg(K = 0) \wedge (L = 0) \wedge \neg(M = 0) \quad \rightarrow \quad \neg(R = 0) \wedge (S = 0)$$

is observed, the *Reasoner* sends to the *CMS* the clauses in SD together with the *input* $\{\neg(K = 0),\ (L = 0),\ \neg(M = 0)\}$. Subsequently, the *Reasoner* queries for the cause based minimal supports for the *output* "$\neg(R = 0) \wedge (S = 0)$". The negation of the cause based minimal supports shown below are the explanations for OBS:

$$\begin{aligned}
\{\quad & \neg ab(X_2) \wedge \neg ab(O_1) \wedge ab(X_1) \wedge \neg ab(A_2) \wedge \neg ab(A_1),\\
& \neg ab(X_2) \wedge ab(X_1) \wedge ab(O_1) \wedge ab(A_1),\\
& \neg ab(X_2) \wedge ab(X_1) \wedge ab(O_1) \wedge ab(A_2),\\
& ab(X_2) \wedge \neg ab(O_1) \wedge ab(A_2) \wedge \neg ab(X_1) \wedge \neg ab(A_1),\\
& ab(X_2) \wedge \neg ab(A_2) \wedge \neg ab(X_1) \wedge ab(O_1),\\
& ab(X_2) \wedge \neg ab(X_1) \wedge ab(O_1) \wedge ab(A_1)\ \}.
\end{aligned}$$

The following table shows the corresponding faulty ($ab(\_)$) and normal ($\neg ab(\_)$) components in each explanation:

	faulty			normal			
1	X_1			X_2	O_1	A_2	A_1
2	X_1	O_1	A_1	X_2			
3	X_1	O_1	A_2	X_2			
4	X_2	A_2		O_1	X_1	A_1	
5	X_2	O_1		A_2	X_1		
6	X_2	O_1	A_1	X_1			

Table 1: The diagnoses for $\neg(R = 0) \wedge (S = 0)$.

Notice that in row (5), the gate A_1 does not have a status. Regardless of the status of A_1, if gates X_2 and O_1 are faulty and gates A_2 and X_1 are normal, then the observation is explained. Also, to emulate the result of (Reiter, 1987) shown earlier i.e., minimizing the faulty components, the minimal explanations are filtered according to only faulty components. That is, restricting to the faulty column, the rows are extracted and subsumption is applied to yield $\{X_1\}$, $\{X_2, A_2\}$ and $\{X_2, O_1\}$. Conversely, maximizing the set of normal components that explain the observation can be obtained similarly. First, the rows under the normal column are extracted and perform the inverse of sumbsumption i.e., no superset of a row is in another row. In the above table, the maximal subsets of normal components that explain the observations are $\{X_2, O_1, A_2, A_1\}$, $\{O_1, X_1, A_1\}$ and $\{A_2, X_1\}$.

Note that it is not necessarily the case that the maximal subset of normal components is the difference between the set of all components and the set of faulty components. The minimal explanation in row (5) is such an example. In general, there are potentially many possible types of causes and whether to maximize or minimize the causes is domain dependent. Thus, the task of extracting preferred causes is performed by the *Reasoner* and the task of the *CMS* is to return all the minimal explanations for each query.

Finally, we assume that the same measurement as in example 7.2 was performed i.e., point P was measured and found to have value 1. To compute the new diagnoses, the *Reasoner* queries the *CMS* for the cause based minimal supports of the new *output* — "$\neg(R=0) \wedge (S=0) \wedge \neg(P=0)$". The negations of these cause based minimal supports

$$\begin{aligned} \{ \quad & \neg ab(X_2) \wedge ab(O_1) \wedge ab(X_1) \wedge ab(A_2), \\ & ab(X_2) \wedge ab(O_1) \wedge \neg ab(A_2) \wedge \neg ab(X_1) \ \} \end{aligned}$$

constitute the minimal explanations for the new observation

$$\neg(K=0) \wedge (L=0) \wedge \neg(M=0) \rightarrow \neg(R=0) \wedge (S=0) \wedge \neg(P=0)$$

with respect to SD.

8 Conclusion

In this paper we have presented a supplementary system for aiding reasoning systems called the *Clause Management System (CMS).* The *CMS* supplements the *Reasoner* by generating explanations for a given query with respect to the knowledge the *Reasoner* sends to the *CMS* . To accomplish this task, the *CMS* relies heavily on the concept of implicates of a set of clauses. We have distinguished three important kinds of implicates: minimal, prime and minimal trivial implicates. We have argued that prime implicates are most important, and that the *CMS* should represent its knowledge base by the set of prime implicates of the clauses it receives from the *Reasoner* , instead of the clauses themselves. An incremental algorithm that updates the *CMS* knowledge base when a new clause is received was also presented.

Subsequently, the notions of minimal, prime and minimal trivial supports for a single clause were introduced and the algorithms to compute them were discussed. We then generalized these algorithms to compute these supports for a finite conjunction of clauses. The latter enables the *CMS* to compute support for any propositional formula. In addition we defined a preference ordering on the supports that gives some ground for further discrimination among the same type supports for a formula.

Finally, we highlighted the functionality of the *CMS* by choosing two logic based diagnostic reasoning paradigms: the consistency and explanation based paradigms. As for future research, the *Reasoner* - *CMS* protocol which is highly domain dependent was not addressed in this paper. We intend to study such protocols for some well known types of reasoning systems in our future work. For example, to investigate the corporation bewteen consistency techniques (Mackworth, 1977) and the *CMS* in the realm of constraint satisfaction problems; and to study the protocol for the 1st order *Reasoner* and the *CMS* in the theorem proving environment. Among other future work, there is the study of the *CMS* in aiding assumption based reasoning systems (Tsiknis & Kean, 1989) for such tasks as nonmonotonic reasoning. Also stated as future research is the investigation into the correspondence bewteen minimal trivial supports and minimal models for a formula. The reward of this study is the extension of the capability of the *CMS* in aiding reasoning systems such as the logical system for depiction and map interpretation (Reiter & Mackworth, 1988).

We are very grateful to Raymond Reiter, Paul Gilmore, Alan Mackworth, Jane Mulligan and David Poole for their comments and criticism.

References

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

Bibel, W., *Automated Theorem Proving (2nd edition)*, Vieweg, Braunschweig, (1987).

Chandra, A.K. and Markowsky, G., On the Number of Prime Implicants, *Discrete Mathematics* **24** (1978) 7-11.

Cox, P.T. and Pietrzykowski, T., Causes for Events: Their Computations and Applications, *Proceeding 8th Conference on Automated Deduction*, Oxford, England (1986) 608-621.

de Kleer, J., An Assumption-based TMS, *Artificial Intelligence* **28** (1986) 127-162.

de Kleer, J. and Williams, B., Diagnosing Multiple Faults, *Artificial Intelligence* **32** (1987) 97-130.

Doyle, J., A Truth Maintenance System, *Artificial Intelligence* **12** (1979) 231-272.

Hwa, H.R., A Method for Generating Prime Implicants of a Boolean Expression, *IEEE Trans. on Computers* (June 1974) 637-641.

Kean, A. and Tsiknis, G., An Incremental Method for Generating Prime Implicants/Implicates, submitted to the *Journal of Symbolic Computation*, also in TR 88-16, Department of Computer Science, University of British Columbia (1988).

Levin, M., Mathematical Logic for Computer Scientists, MAC TR-131, MIT, (1974).

Mackworth, A.K., Consistency in Networks of Relations, *Artificial Intelligence 8* (1977) 99-118.

Milne, R., Strategies for Diagnosis, *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-**17/3** (May/June 1987).

Morgan, C.G., Hypothesis Generation by Machine, *Artificial Intelligence* **2** (1971) 179-187.

Poole, D., Goebel, R. and Aleliunas, R., Theorist: A logical Reasoning System for Defaults and Diagnosis, Research Report CS-86-06, Logic Programming and Artificial Intelligence Group, Department of Computer Science, University of Waterloo, (1986).

Poole, D., Explanation and Prediction: An Architecture for Default and Abductive Reasoning, Logic Programming and Artificial Intelligence Group, Department of Computer Science, University of Waterloo, (1988).

Poole, D., Representing Knowledge for Logic-based Diagnosis, *Proceedings of International Conference on Fifth Generation Computing*, Tokyo, (Nov 1988).

Poole, D., Diagnosis with Fault Models, Logic Programming and Artificial Intelligence Group, Department of Computer Science, University of Waterloo, (1988).

Reiter, R., A Theory of Diagnosis from First Principle, *Artificial Intelligence* **32** (1987) 57-95.

Reiter, R. and de Kleer, J., Foundations of Assumption-Based Truth Maintenance Systems: Preliminary Report, *Proceeding AAAI-87*, Seatle, Washington, (1987) 183-188.

Reiter, R. and Mackworth, A.K., A Logical Framework for Depiction and Image Interpretation, TR 88-17, Department of Computer Science, University of British Columbia, (1988).

Shoenfield, J.R., *Mathematical Logic*, Addison-Wesley, (1967).

Slagle, J.R., Chang, C.L. and Lee, R.C.T., A New Algorithm for Generating Prime Implicants, *IEEE Trans. on Computers*, Vol. **C-19/4** (Apr 1970).

Tison, P., Generalized Consensus Theory and Application to the Minimization of Boolean Functions, *IEEE Transaction on Electronic Computers*, EC-**16/4** (Aug 1967) 446-456.

Tsiknis, G. and Kean, A. Assumption Based Reasoning and Clause Management Systems, in preparation, (1989).

Appendix

The following is the complete sets of formulae for the explanation-based approach in diagnostic reasoning (see figure 6). The causes $\mathcal{C}$ are the set of all $ab(U)$ for U denoting all the five gates in the full adder. The system description is a set of formulae describing the *causes* and *effects* of the gates for all the possible combination of observations.

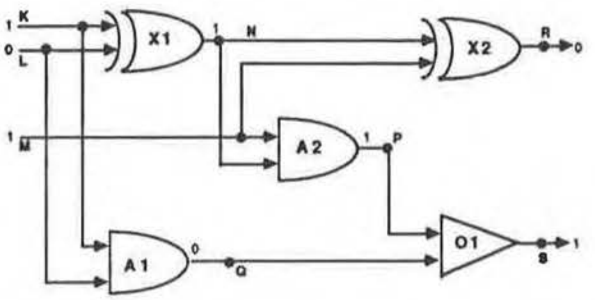


Figure 6: A Full Adder

$\mathcal{C} = \{ab(X_1),\ ab(X_2),\ ab(A_1),\ ab(A_2),\ ab(O_1)\}$

$SD = \{$

$\neg(K=0) \lor \neg(L=0) \lor ab(X_1) \lor (N=0),$
$\neg(K=0) \lor (L=0) \lor ab(X_1) \lor \neg(N=0),$
$(K=0) \lor \neg(L=0) \lor ab(X_1) \lor \neg(N=0),$
$(K=0) \lor (L=0) \lor ab(X_1) \lor (N=0),$

$\neg(K=0) \lor \neg(L=0) \lor \neg ab(X1) \lor \neg(N=0),$
$\neg(K=0) \lor (L=0) \lor \neg ab(X1) \lor (N=0),$
$(K=0) \lor \neg(L=0) \lor \neg ab(X1) \lor (N=0),$
$(K=0) \lor (L=0) \lor \neg ab(X1) \lor \neg(N=0),$

$(K=0) \lor (L=0) \lor ab(A_1) \lor \neg(Q=0),$
$\neg(K=0) \lor ab(A_1) \lor (Q=0),$
$\neg(L=0) \lor ab(A_1) \lor (Q=0),$

$(K=0) \lor (L=0) \lor \neg ab(A1) \lor (Q=0)$
$\neg(K=0) \lor \neg ab(A1) \lor \neg(Q=0),$
$\neg(L=0) \lor \neg ab(A1) \lor \neg(Q=0),$

$\neg(P=0) \lor \neg(Q=0) \lor ab(O_1) \lor (S=0),$
$(P=0) \lor ab(O_1) \lor \neg(S=0),$
$(Q=0) \lor ab(O_1) \lor \neg(S=0)$

$\neg(N=0) \lor \neg(M=0) \lor ab(X_2) \lor (R=0),$
$\neg(N=0) \lor (M=0) \lor ab(X_2) \lor \neg(R=0),$
$(N=0) \lor \neg(M=0) \lor ab(X_2) \lor \neg(R=0),$
$(N=0) \lor (M=0) \lor ab(X_2) \lor (R=0),$

$\neg(N=0) \lor \neg(M=0) \lor \neg ab(X2) \lor \neg(R=0),$
$\neg(N=0) \lor (M=0) \lor \neg ab(X2) \lor (R=0),$
$(N=0) \lor \neg(M=0) \lor \neg ab(X2) \lor (R=0),$
$(N=0) \lor (M=0) \lor \neg ab(X2) \lor \neg(R=0),$

$(M=0) \lor (N=0) \lor ab(A_2) \lor \neg(P=0),$
$\neg(M=0) \lor ab(A_2) \lor (P=0),$
$\neg(N=0) \lor ab(A_2) \lor (P=0),$

$(M=0) \lor (N=0) \lor \neg ab(A2) \lor (P=0),$
$\neg(M=0) \lor \neg ab(A2) \lor \neg(P=0),$
$\neg(N=0) \lor \neg ab(A2) \lor \neg(P=0),$

$\neg(P=0) \lor \neg(Q=0) \lor \neg ab(O1) \lor \neg(S=0),$
$(P=0) \lor \neg ab(O1) \lor (S=0),$
$(Q=0) \lor \neg ab(O1) \lor (S=0)\ \}$

Assuming that the *Reasoner* has observed the following observations:

input $= \{\neg(K=0),\ L=0,\ \neg(M=0)\}$

$output = \{\neg(R = 0),\ S = 0\}$
$OBS = input \rightarrow output$

The *Reasoner* transmits to the *CMS* the *input* and the *CMS* computes the set of prime implicates for $SD \cup input$ as shown below.

$PI(SD \cup input) = \{$

$(L = 0),$
$\neg(M = 0),$
$\neg ab(A1) \vee \neg(Q = 0),$
$ab(A1) \vee (Q = 0),$
$ab(A2) \vee \neg(P = 0) \vee ab(X1),$
$\neg(N = 0) \vee \neg ab(X2) \vee (R = 0),$
$\neg(N = 0) \vee ab(X2) \vee \neg(R = 0),$
$(P = 0) \vee ab(O1) \vee \neg(S = 0),$
$(Q = 0) \vee ab(O1) \vee \neg(S = 0),$
$\neg ab(X1) \vee \neg ab(A2) \vee \neg(P = 0),$
$ab(X2) \vee (R = 0) \vee ab(X1),$
$\neg ab(X2) \vee \neg(R = 0) \vee ab(X1),$
$ab(O1) \vee \neg(S = 0) \vee \neg ab(A1),$
$\neg ab(X1) \vee ab(A2) \vee (P = 0),$
$(N = 0) \vee \neg ab(A2) \vee (P = 0),$
$\neg(P = 0) \vee \neg(Q = 0) \vee ab(O1) \vee (S = 0),$
$ab(A2) \vee \neg(P = 0) \vee \neg ab(X2) \vee (R = 0),$
$\neg ab(X2) \vee \neg(R = 0) \vee \neg ab(A2) \vee \neg(P = 0),$
$\neg(P = 0) \vee \neg ab(O1) \vee \neg(S = 0) \vee ab(A1),$
$(N = 0) \vee ab(A2) \vee \neg ab(O1) \vee (S = 0),$
$ab(A2) \vee ab(X1) \vee \neg ab(O1) \vee (S = 0),$
$ab(O1) \vee \neg(S = 0) \vee \neg(N = 0) \vee \neg ab(A2),$
$\neg ab(X1) \vee \neg ab(O1) \vee (S = 0) \vee \neg ab(A2),$
$\neg ab(X2) \vee \neg(R = 0) \vee ab(A2) \vee (P = 0),$
$\neg ab(A2) \vee (P = 0) \vee \neg ab(X2) \vee (R = 0),$
$ab(O1) \vee (S = 0) \vee \neg ab(A2) \vee ab(X1) \vee ab(A1),$
$ab(O1) \vee (S = 0) \vee \neg ab(X1) \vee ab(A2) \vee ab(A1),$
$ab(O1) \vee (S = 0) \vee \neg(N = 0) \vee ab(A2) \vee ab(A1),$
$\neg(Q = 0) \vee ab(O1) \vee (S = 0) \vee (N = 0) \vee \neg ab(A2),$
$\neg(Q = 0) \vee ab(O1) \vee (S = 0) \vee \neg ab(A2) \vee ab(X1),$
$\neg(Q = 0) \vee ab(O1) \vee (S = 0) \vee \neg ab(X1) \vee ab(A2),$
$\neg(Q = 0) \vee ab(O1) \vee (S = 0) \vee \neg(N = 0) \vee ab(A2),$
$\neg(Q = 0) \vee \neg ab(O1) \vee \neg(S = 0) \vee \neg(N = 0) \vee ab(A2),$
$\neg(Q = 0) \vee \neg ab(O1) \vee \neg(S = 0) \vee \neg ab(X1) \vee ab(A2),$
$\neg(Q = 0) \vee \neg ab(O1) \vee \neg(S = 0) \vee \neg ab(A2) \vee ab(X1),$
$\neg(Q = 0) \vee \neg ab(O1) \vee \neg(S = 0) \vee (N = 0) \vee \neg ab(A2),$
$\neg ab(O1) \vee \neg(S = 0) \vee \neg(N = 0) \vee ab(A2) \vee ab(A1),$
$\neg ab(O1) \vee \neg(S = 0) \vee \neg ab(X1) \vee ab(A2) \vee ab(A1),$
$\neg ab(O1) \vee \neg(S = 0) \vee \neg ab(A2) \vee ab(X1) \vee ab(A1),$
$\neg ab(O1) \vee \neg(S = 0) \vee (N = 0) \vee \neg ab(A2) \vee ab(A1),$
$ab(A2) \vee ab(X2) \vee \neg(R = 0) \vee ab(O1) \vee \neg(S = 0),$
$ab(A2) \vee ab(X2) \vee \neg(R = 0) \vee \neg ab(O1) \vee (S = 0),$
$ab(A2) \vee \neg ab(X2) \vee (R = 0) \vee ab(O1) \vee \neg(S = 0),$
$ab(A2) \vee \neg ab(X2) \vee (R = 0) \vee \neg ab(O1) \vee (S = 0),$
$ab(X2) \vee (R = 0) \vee \neg ab(O1) \vee (S = 0) \vee \neg ab(A2),$
$ab(X2) \vee (R = 0) \vee ab(O1) \vee \neg(S = 0) \vee \neg ab(A2),$

$\neg(K = 0),$
$ab(X1) \vee \neg(N = 0),$
$\neg ab(X1) \vee (N = 0),$
$(N = 0) \vee ab(A2) \vee \neg(P = 0),$
$\neg ab(X2) \vee (R = 0) \vee \neg ab(X1),$
$ab(X2) \vee \neg(R = 0) \vee \neg ab(X1),$
$(P = 0) \vee \neg ab(O1) \vee (S = 0),$
$\neg(N = 0) \vee \neg ab(A2) \vee \neg(P = 0),$
$(Q = 0) \vee \neg ab(O1) \vee (S = 0),$
$(N = 0) \vee ab(X2) \vee (R = 0),$
$(N = 0) \vee \neg ab(X2) \vee \neg(R = 0),$
$\neg(N = 0) \vee ab(A2) \vee (P = 0),$
$\neg ab(O1) \vee (S = 0) \vee \neg ab(A1),$
$\neg ab(A2) \vee (P = 0) \vee ab(X1),$
$\neg(P = 0) \vee ab(O1) \vee (S = 0) \vee ab(A1),$
$ab(A2) \vee \neg(P = 0) \vee ab(X2) \vee \neg(R = 0),$
$ab(X2) \vee (R = 0) \vee \neg ab(A2) \vee \neg(P = 0),$
$\neg(P = 0) \vee \neg(Q = 0) \vee \neg ab(O1) \vee \neg(S = 0),$
$(N = 0) \vee ab(A2) \vee ab(O1) \vee \neg(S = 0),$
$ab(A2) \vee ab(X1) \vee ab(O1) \vee \neg(S = 0),$
$\neg ab(O1) \vee (S = 0) \vee \neg(N = 0) \vee \neg ab(A2),$
$\neg ab(X1) \vee ab(O1) \vee \neg(S = 0) \vee \neg ab(A2),$
$ab(X2) \vee (R = 0) \vee ab(A2) \vee (P = 0),$
$\neg ab(A2) \vee (P = 0) \vee ab(X2) \vee \neg(R = 0),$
$ab(O1) \vee (S = 0) \vee (N = 0) \vee \neg ab(A2) \vee ab(A1),$

$\neg ab(X2) \lor \neg(R=0) \lor \neg ab(O1) \lor (S=0) \lor \neg ab(A2),$
$\neg ab(X2) \lor \neg(R=0) \lor ab(O1) \lor \neg(S=0) \lor \neg ab(A2),$
$ab(O1) \lor (S=0) \lor \neg ab(A2) \lor \neg ab(X2) \lor (R=0) \lor ab(A1),$
$ab(O1) \lor (S=0) \lor \neg ab(A2) \lor ab(X2) \lor \neg(R=0) \lor ab(A1),$
$ab(O1) \lor (S=0) \lor \neg ab(X2) \lor \neg(R=0) \lor ab(A2) \lor ab(A1),$
$ab(O1) \lor (S=0) \lor ab(X2) \lor (R=0) \lor ab(A2) \lor ab(A1),$
$\neg(Q=0) \lor ab(O1) \lor (S=0) \lor \neg ab(A2) \lor \neg ab(X2) \lor (R=0),$
$\neg(Q=0) \lor ab(O1) \lor (S=0) \lor \neg ab(A2) \lor ab(X2) \lor \neg(R=0),$
$\neg(Q=0) \lor ab(O1) \lor (S=0) \lor \neg ab(X2) \lor \neg(R=0) \lor ab(A2),$
$\neg(Q=0) \lor ab(O1) \lor (S=0) \lor ab(X2) \lor (R=0) \lor ab(A2),$
$\neg(Q=0) \lor \neg ab(O1) \lor \neg(S=0) \lor ab(X2) \lor (R=0) \lor ab(A2),$
$\neg(Q=0) \lor \neg ab(O1) \lor \neg(S=0) \lor \neg ab(X2) \lor \neg(R=0) \lor ab(A2),$
$\neg(Q=0) \lor \neg ab(O1) \lor \neg(S=0) \lor \neg ab(A2) \lor ab(X2) \lor \neg(R=0),$
$\neg(Q=0) \lor \neg ab(O1) \lor \neg(S=0) \lor \neg ab(A2) \lor \neg ab(X2) \lor (R=0),$
$\neg ab(O1) \lor \neg(S=0) \lor ab(X2) \lor (R=0) \lor ab(A2) \lor ab(A1),$
$\neg ab(O1) \lor \neg(S=0) \lor \neg ab(X2) \lor \neg(R=0) \lor ab(A2) \lor ab(A1),$
$\neg ab(O1) \lor \neg(S=0) \lor \neg ab(A2) \lor ab(X2) \lor \neg(R=0) \lor ab(A1),$
$\neg ab(O1) \lor \neg(S=0) \lor \neg ab(A2) \lor \neg ab(X2) \lor (R=0) \lor ab(A1)\ \}$

In the explanation based approach, the diagnoses are the minimal supports for the *output* that consist solely of *causes* as shown below.

$output = \{\neg(R=0), (S=0)\}$
the set of minimal supports for *output* w.r.t. $SD \cup input$ is:
$\{\ ab(X2) \lor ab(O1) \lor \neg ab(X1) \lor ab(A2) \lor ab(A1),$
$ab(X2) \lor \neg ab(X1) \lor \neg ab(O1) \lor \neg ab(A1),$
$ab(X2) \lor \neg ab(X1) \lor \neg ab(O1) \lor \neg ab(A2),$
$\neg ab(X2) \lor ab(O1) \lor \neg ab(A2) \lor ab(X1) \lor ab(A1),$
$\neg ab(X2) \lor ab(A2) \lor ab(X1) \lor \neg ab(O1),$
$\neg ab(X2) \lor ab(X1) \lor \neg ab(O1) \lor \neg ab(A1)\ \}.$

After examing the diagnoses for the *output* given above, the *Reasoner* proceeds to test the circuit and discovered that the wire P has a non-zero value. The *Reasoner* queries the *CMS* with the new information and obtained the following diagnoses.

$output = \{\neg(R=0), (S=0), \neg(P=0)\}$
the set of minimal supports for the new *output* w.r.t. $SD \cup input$ is:
$\{\ ab(X2) \lor \neg ab(O1) \lor \neg ab(X1) \lor \neg ab(A2),$
$\neg ab(X2) \lor \neg ab(O1) \lor ab(A2) \lor ab(X1)\ \}.$