SOLVING DIAGNOSTIC PROBLEMS USING EXTENDED ASSUMPTION-BASED TRUTH MAINTENANCE SYSTEMS: FOUNDATIONS

by

Gregory M. Provan

Technical Report 88-10

SOLVING DIAGNOSTIC PROBLEMS USING EXTENDED ASSUMPTION-BASED TRUTH MAINTENANCE SYSTEMS: FOUNDATIONS

Gregory M. Provan[1]

Technical Report 88-10

July 1988

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

Abstract

We describe the use of efficient, extended Truth Maintenance Systems (TMSs) for diagnosis. We show that for complicated diagnostic problems, existing ATMSs need some method of ranking competing explanations and pruning the search space in order to maintain computational efficiency. We describe a specific implementation of an ATMS for efficient problem solving that incorporates the full Dempster Shafer theory in a semantically clear and efficient manner. Such an extension allows the Problem Solver to rank competing solutions and explore only the "most likely" solutions. We also describe several efficient algorithms for computing both exact and approximate values for Dempster Shafer belief functions.

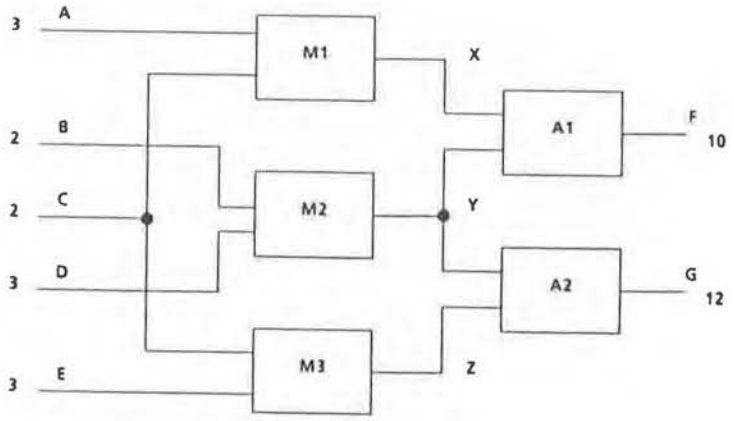
[1]The author completed this research with the support of a Scholarship from the Rhodes Trust, Oxford, and of the University of British Columbia Center for Integrated Computer Systems Research, BC Advanced Systems Institute and NSERC grants to A.K. Mackworth.

1 Introduction

1.1 Diagnostic Problems

Diagnostic reasoning is the process of inferring a set of best explanations for a body of evidence. It may be thought of as the dual to the deductive process of finding the set D of facts derivable from a set of facts X, i.e. $g(X) = D$. In other words, we want some inverse function g^{-1} such that $g^{-1}(D) = X$. We call such a definition of diagnosis *functional*, as D is a function of X, or in other terminology D is entailed by X, $X \models D$. We call a non-functional form of diagnosis *consistency-based*. Here, the functional or entailment relationships do not necessarily hold: $g(X) \neq D$, or $X \not\models D$. Instead, we have a weaker, consistency-based notion $f(X) \bowtie D$, meaning that D is consistent with X, i.e. you cannot derive the negation of any clauses in D. There are well-known techniques for the primal (deduction) problem, such as resolution ([8] etc.), but the dual techniques are less well-researched. This $g^{-1}(D) = X$ paradigm covers a wide range of problems in Artificial Intelligence. For example, it covers as seemingly disparate areas as circuit diagnosis and computer vision. As an example of a typical diagnostic problem, consider the well-known circuit consisting of multipliers M_1, M_2 and M_3 and adders A_1 and A_2, as shown in Figure 1. The output at F is 10 instead of 12, meaning that some combination(s)

Figure 1: Circuit with faulty components



of M_1, M_2, M_3, A_1 and A_2 is(are) faulty. There are a total of $2^5 = 32$ combinations of faulty components. Using a consistency-based approach, taking observations at points like X, Y or Z narrows the set of diagnoses consistent with the observations and guides future decisions about where to make further readings. A solution consists of a set of faulty multipliers and adders consistent with all the observations.

As a second example, consider the high level vision problem of generating 3D interpretations from 2D line drawings. Taking a functional approach, the task, roughly speaking, is to identify the objects in a scene which have caused a given pattern of lines in an image. In a sense, the task is to "diagnose" the image as being caused by some set of objects. The primal problem, i.e. $g(X) = D$, is a computer graphics problem, namely to generate a 2D projection of a set of 3D objects in some orientation. Well-known, efficient techniques exist for this task for any objects and orientations. The dual high level vision problem, which we defined earlier, is significantly more difficult, and at present is limited in the type of objects which can be identified.

There have been several approaches to diagnosis, in fields such as circuit design ([11], [14], [12]) and medicine ([37], [42]). Two important goals of any diagnostic system are efficiency and the ability to generate the best explanations (of which there may be multiple interacting ones) for the observed symptoms. These two goals conflict with one another, as the more powerful the diagnostic system (e.g. generating all explanations vs. a single one) the more computationally expensive it becomes.

We argue that diagnostic systems are most efficient when (1) they incorporate *both* numeric and non-numeric solution methods, and (2) use a heuristic decision procedure to guide the explanation-generation (i.e. prune the number of possible partial explanations), rather than enumerating *all* potential explanations and then optimizing over the explanation set. Many model-based reasoning systems generate all possible explanations before selecting the "optimal" explanations, e.g. [14], [37]. However, although the causal approach enables a more complete description of the causes for a body of evidence, it is computationally less efficient than the evidential approach, especially if it uses binary weights (or uncertainty representation scheme). For complex reasoning tasks, such an approach can be computationally infeasible because of the large search space which may be generated [32]. Pruning unpromising solutions using a numerical weighting system and an appropriate heuristic for defining "optimality" can improve efficiency and ensure that the optimal solutions are always found.

1.2 Overview

We describe a specific implementation of a problem-solving tool, the Assumption-based TMS (ATMS), suitable for complex diagnostic problems. This extended ATMS computes Dempster-Shafer (DS) mass functions for hypothesised diagnoses without compromising the semantic clarity or computational efficiency of present ATMSs, e.g. [13]. Existing ATMSs cannot rank hypotheses since they are limited to binary truth values. The incorporation of DS mass functions allows hypotheses to be ranked and offers additional efficiency improvements, such as the ability to prune the search space by identifying highly unlikely partial solutions. We then show that this extended ATMS is a semantically clear system which is more efficient than existing diagnostic uses of the ATMS, such as the General

Diagnostic Engine (GDE) [14] and GMODS [21].

This paper is organised as follows. In Section 2 we introduce the terminology and definitions used in the rest of the paper. In specific, we describe existing diagnostic uses of the ATMS, showing the drawbacks of such systems. We then review the theoretical basis of the ATMS and DS theory. We also discuss previous implementations of DS theory, and note that, as opposed to this implementation of the *full* DS theory, all previous implementations were of restrictions of DS theory because of the intractability of computing DS weights. In Section 3 we describe the semantic correspondence between the ATMS and DS theory. We give a small example of how DS weights can be computed from ATMS labels. In Section 4 we briefly outline the implementation of an extended ATMS for diagnosis.

2 Preliminaries

2.1 Existing Diagnostic Uses of TMSs

Truth Maintenance Systems (TMSs) are an AI reasoning tool which have been used for a variety of applications. We represent the input alphabet (using terminology introduced in [38]) by a set $X = \{x_1, ..., x_n\}$ of propositional symbols. A propositional literal is a propositional symbol or its negation. A TMS records dependencies among a set of facts to maintain a consistent (set of) valuation(s) for the fact set as the fact set changes over time. A fact is a propositional Horn clause consisting of a propositional literal or a finite disjunction of propositional literals with no literal repeated, e.g. $\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee ... \vee \overline{x_k} \vee x$, $k \geq 0$.[2] A valuation is a truth assignment (believed or not believed) with respect to a given fact set truth assignment and set of dependency records. x_i is dependent on x_j if the valuation of x_i changes as the valuation of x_j changes. A TMS uses such dependency information to rule out regions of the search space and to efficiently indicate necessary database changes when contradictions are discovered.

There are two types of TMSs, Justification-based TMSs (JTMS) (e.g. [27], [15]) and Assumption-Based TMSs (ATMS) (e.g. [13]). The ATMS has received the most attention recently, and has proven to be a powerful tool for diagnosis, since it enables multiple interacting diagnoses to be identified, can perform incremental diagnoses, and records the complete problem structure, among other capabilities. One such application, GDE [14], can simultaneously find all sets of faulty circuit components which explain the observations for circuits as in Figure 1. In GDE, the ATMS identifies hypothesised sets of circuit components whose faulty behaviour could cause discrepancies between predicted and observed circuit measurements. Entropy values are assigned to these faulty component sets to guide future measurements and incrementally narrow the set of hypotheses.

Although GDE offers several advantages over previous diagnostic systems, such as its

[2] We can represent any fact as either $x_i \supset x_j$ or $\overline{x_i} \vee x_j$. We use the latter method by convention.

domain independence and ability to identify all multiple, interacting causes, there are several ways in which it can be improved, of which we mention two. First, a [0,1] uncertainty representation scheme within the ATMS[3] can allow a context pruning mechanism to prune all but the most likely contexts. This avoids a possible combinatorial explosion of the number of contexts caused by GDE's computation of all possible diagnoses. Such a combinatorial explosion can occur when diagnosing complex circuits, due to the ATMS needing to create a huge number of contexts to represent a large number of partial diagnoses, as shown by Provan [32]. Hence, some pruning mechanism is needed to avoid this combinatorial explosion. The binary weights in an ATMS cannot rank contexts representing partial solutions, e.g. in terms of a criterion such as "likelihood to generate a complete solution". Second, for complex circuits computational inefficiency can occur because GDE must repeatedly generate the current set of diagnoses (called candidates) from inconsistencies in the circuit (conflict sets), a computationally expensive process. It would be better to generate the diagnoses directly.

2.2 ATMS Review

The ATMS records dependencies in terms of a distinguished subset $A \subseteq X$ of literals called *assumptions*. We refer to an *environment* E as a set of assumptions and a *context* C as the set of literals derivable from the assumptions in E given the facts.

The input consists of a set $X = \{X_1, ..., X_l\}$ of *facts*, referred to in [13] as justifications. We call the conjunction of the X_i's a Boolean polynomial function[4] F, i.e. $F = \bigwedge_{i=1,...,l} X_i$. We note that there may be many other polynomials F' which also compute such a partially specified Boolean function F, where a polynomial F' computes F if $F'(x) = F(x)$ for every instantiation of $\mathbf{x}$.

Assumptions are the fundamental literals with which the derivation of all other literals are recorded. For example, in the network diagnosis problem described in Section 1.1, assumptions can be: (1) each component is working, represented by M_1, M_2, etc. in Figure 1; or (2) input values, e.g. $A = 3$, $B = 2$, etc.

Facts containing an unnegated literal, e.g. $\overline{x_1} \vee \overline{x_3} \vee x_5$, have an antecedent (which can be a set of assumptions, as in $\overline{x_1} \vee \overline{x_3}$) justifying a consequent x_5, which cannot be an assumption. We also call a derived literal the consequent of a fact. In the network diagnosis example, facts could be as follows:

$$\overline{M_1} \vee (X = 6)$$
$$\overline{M_2} \vee (Y = 6)$$
$$\overline{A_1} \vee (F = 6).$$

[3] In GDE, the ATMS and the uncertainty representation module (which uses an entropy method) are separate.

[4] This is standard conjunctive normal form (CNF), the dual representation of traditional disjunctive normal form (DNF) Boolean functions.

The first of these facts means that if M_1 is working, the reading at point X in the circuit should be 6.

The ATMS's operation consists of two distinct phases, label manipulation and interpretation construction, which we describe below.

Label Manipulation: In the label manipulation phase, for each literal the ATMS maintains a *label*, which is a set of environments in which the literal can be proven. Each environment consists of the greatest lower bound (GLB) of assumption sets. In logical terms, each environment in the label for x_i is defined in terms of a minimal support clause ξ_k^* for x_i, and is given by $\{\bigwedge_j A_j$ for some $j \mid (\bigvee_j \overline{A_j})$ is a minimal support clause of $X\}$, using the definition of [38]. ξ_k is a *support clause* for x_i with respect to X if $X \not\models \xi_k$, $x_i \cup \xi_j$ does not contain a complementary pair of literals (i.e. both x_j and $\overline{x_j}$), and $X \models x_i \cup \xi_k$. ξ_k^* is a minimal support clause for x_i with respect to X if no proper subset of ξ_k is a support clause for x_i with respect to X. A closely related type of clause, a prime implicate,[5] is defined as follows: a prime implicate of a set X of clauses is a clause π such that

- $X \models \pi$, and
- For no proper subset π' of π does $X \models \pi'$.

The relationship between the label (minimal support clause) for a literal x and a prime implicate is given as follows [38]: ξ is a minimal support clause for x with respect to X iff there is a prime implicate π for X such that $x \in \pi$ and $\xi = \pi - x$.

Labels are assigned to a derived literal by taking the set union of all combinations of labels for its antecedents and then using subsumption to ensure the new label is represented in GLB form. Inconsistencies are removed by identifying the inconsistent sets of assumptions (nogoods) and removing these sets and their supersets from all labels.

For example, if we have $\overline{x_1} \vee \overline{x_3} \vee x_5$, and x_1 and x_3 have labels $\{\{A, B\}, \{B, C\}\}$ and $\{\{A\}, \{D, E\}\}$ respectively, then x_5 is assigned the label $\{\{A, B\}, \{B, C, D, E\}\}$.

The ATMS explores multiple solutions simultaneously, implicitly representing each solution with a *context*. A derived literal is contained in a context if the assumptions in at least one of the environments in its label are a subset of that context. Multiple labels for a literal x_i indicate that x_i is present in multiple contexts. However, this procedure does not explicitly calculate contexts, but manipulates labels. Contexts are explicitly computed by an interpretation construction algorithm.

Interpretation Construction: In the second phase of operation, the ATMS constructs the *interpretations* to explicitly determine the set of "solutions" or maximal contexts. An *interpretation* is the smallest set of assumptions from which all literals in the context are derivable, and a context is maximal if it has no consistent superset contexts.

[5] The dual to prime implicate (in Boolean algebra) is called a prime implicant. We use the prime implicate terminology to avoid confusion between the dual representations.

In logical terminology, the ATMS computes the *minimal* or *irredundant* polynomial $\mathcal{F}$ for F. If we call assign a unit cost to each support clause ξ_i, then we define the cost of F as the sum of the costs of the support clauses ξ_i in F.[6] A minimal polynomial $\mathcal{F}$ is a polynomial such that $\mathcal{F}$ computes F and no polynomial computing F has cost smaller than $\mathcal{F}$.

2.3 Dempster-Shafer Theory Review

Many good descriptions of Dempster-Shafer (DS) theory exist, e.g. [29], [30], [40], [43]. We state a few basic relationships, and refer the reader to the references.

In DS theory, weights are assigned to subsets as well as elements of the set of propositions. A mass function ϱ assigns weights to proposition θ and proposition set (or frame of discernment) Θ subject to the following properties: $\varrho(\theta) \in [0,1]$, $\sum_{\theta \in \Theta} \varrho(\theta) = 1$ and $\varrho(\emptyset) = 0$.

There are two measures in DS theory which are derived from this mass function: *Belief*, the degree of belief in proposition subsets from which θ can be proven:

$$Bel(\theta) = \sum_{\varphi \subseteq \theta} \varrho(\varphi), \tag{1}$$

and *Plausibility*, the belief in subsets that do not disprove θ:

$$Pls(\theta) = 1 - \sum_{\varphi \subseteq \neg\theta} \varrho(\varphi) = 1 - Bel(\neg\theta). \tag{2}$$

These two measures can be used as upper and lower bounds on the preference order:

$$\forall \theta \in \Theta, \quad Bel(\theta) \leq \varrho(\theta) \leq Pls(\theta). \tag{3}$$

Dempster's Rule of Combination defines an updated belief function for a proposition θ provable in terms of θ_i and θ_j as:

$$\varrho'(\theta) = \frac{\sum_{i,j:\theta_i \cap \theta_j = \theta} \varrho(\theta_i)\varrho(\theta_j)}{1 - \sum_{i,j:\theta_i \cap \theta_j = \emptyset} \varrho(\theta_i)\varrho(\theta_j)}. \tag{4}$$

The numerator assumes independence of propositions.[7] Viewed in set-theoretic terms, this is simply "summing" the mass functions of all sets in which θ is provable. The denominator of Equation 4 is a normalizing term, given that DS belief is assigned only to non-contradictory subsets.

[6] There are many ways to define such a cost function. For example, we can assign the cost of each support clause ξ_i as the number of assumptions in ξ_i.

[7] All approaches to reasoning with uncertainty, e.g. Bayes nets, etc. must make independence assumptions of one sort or another for computational tractability.

2.4 Related Implementations of Dempster-Shafer Theory

Aside from recent ATMS implementations, only restrictions of DS theory have previously been implemented because of the computational complexity associated with computing mass functions. The number of subsets of Θ increases exponentially with $|\Theta|$, and the normalizing function can sum over all of these subsets, so computing a single normalization function can be computationally expensive. The total space necessary to compute DS belief functions over a set of n propositions is 2^{2^n}.

Examples of such restricted implementations include work by Shafer and Logan and by d'Ambrosio. Shafer and Logan [41] have implemented a restriction based on a proposal by Barnett [6], as well as a restriction based on a proposal by Gordon and Shortliffe [19]. D'Ambrosio [10] has implemented a restricted form of DS theory based on the Support Logic Programming of Baldwin [3]. D'Ambrosio attaches a *simplification* of the Dempster-Shafer uncertainty bounds to ATMS labels. This approach evaluates the DS weights *after* the ATMS has symbolically determined the maximal context sets. The advantages of this approach include the ability to (1) rank solutions, (2) reason with non-independent evidence, and (3) analyse the certainty bound of any literal from multiple perspectives (if it exists in multiple contexts).

We extend the ATMS with DS theory in a manner similar to that of d'Ambrosio. However, unlike d'Ambrosio's method, we use the *full* DS theory, base this extension upon semantic appropriateness (of which d'Ambrosio made no mention), use a different notion of uncertainty primitive, and discuss a more efficient implementation than that proposed by d'Ambrosio.

D'Ambrosio's approach is based upon having the DS mass function as the uncertainty primitive, and ATMS assumptions as proposition tokens (or truth variables). Our approach more closely follows the ATMS's notion of assumption sets being the primitive data representation; hence, DS mass function assignments to the assumption sets summarise the "proofs" denoted by the assumption sets in terms of a single (or pair of) uncertainty value(s).

Laskey and Lehner [23] have independently extended the ATMS with the full DS theory in a manner more or less identical to the extension proposed here and also described in [34]. However, Laskey and Lehner do not discuss the complexity of this DS theory implementation, and their simple implementation of the computation of DS Belief functions from ATMS labels (which we refer to as Network Reliability computation) will be intractable for all but the simplest problems. Here, we discuss both the semantics of extending the ATMS and the efficiency issues surrounding this extension.

Pearl [29] describes this semantic correspondence between the ATMS and DS theory in a manner almost identical to the one presented here. He motivates this correspondence using the analogue of a random switch, which we discuss in the next section. However, he does not provide an implementation of an ATMS for computing DS mass functions.

3 Extension of ATMS with Dempster Shafer Theory

To enable the ATMS to rank solutions and hence improve the ATMS's diagnostic efficiency (e.g. by pruning the search space of possible solutions), we propose assigning a [0,1] uncertainty representation. We want to define an efficient TMS with a clear semantics for solving diagnostic problems. It is not sufficient just to assign an uncertainty representation without the entire system having a clear semantics, as that will obscure the decision-making process. Moreover, introducing an arbitrary weighting system may be counterproductive from a computational point of view, as it may make the resulting diagnostic tool less efficient than existing TMSs.

3.1 Semantic Correspondence of the ATMS and Dempster Shafer Theory

DS theory can be incorporated into the ATMS in a semantically clear and efficient manner as their semantics are identical. To show the consistency of the semantics of the ATMS and DS theory, we need to show a correspondence between

- assumptions and mass functions, i.e. $A_i \Longleftrightarrow \varrho(A_i)$.
- Assumption sets and *Bel* and *Pls* assignments, i.e. $\bigwedge_i A_i \Longleftrightarrow Bel(\bigwedge_i A_i)$, $Pls(\bigwedge_i A_i)$.
- ATMS label updating and DS updating

Assumption/mass function correspondence: De Kleer [13] defines an assumption *not* as a regular literal with a truth assignment, but as a distinguished literal with a *hypothetical* truth assignment. It is very simple to quantise this hypothetical truth value using a DS mass function. Such a quantisation reflects the "confidence" with which the truth value holds.

Pearl [29] describes this notion of quantisation using a random switch model, as shown in Figure 2. The mass function defines the fraction of time a switch assigns a truth value to the assumption. Hence, as in Figure 2, the switch assigns a truth value to assumption A a fraction p of the time, and no assignment $1 - p$ of the time.

Assumption set/Belief and Plausibility correspondence: The ATMS can identify the sets of assumptions from which given literals can be logically proven; hence, if a literal x has label $L = \{L_1, L_2, L_3\}$, it can be proven if any assumption set L_1 or L_2 or L_3 is valid. The ATMS's label for a literal corresponds to the frequency with which that literal can be proven from the initial set of assumptions, i.e. its DS mass function can be computed from the label if a DS mass function is assigned to each assumption.

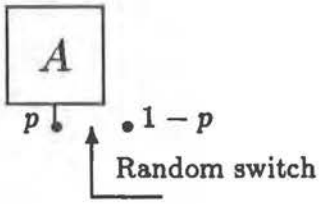


Figure 2: Random switch model for mass function assignment to ATMS assumption.

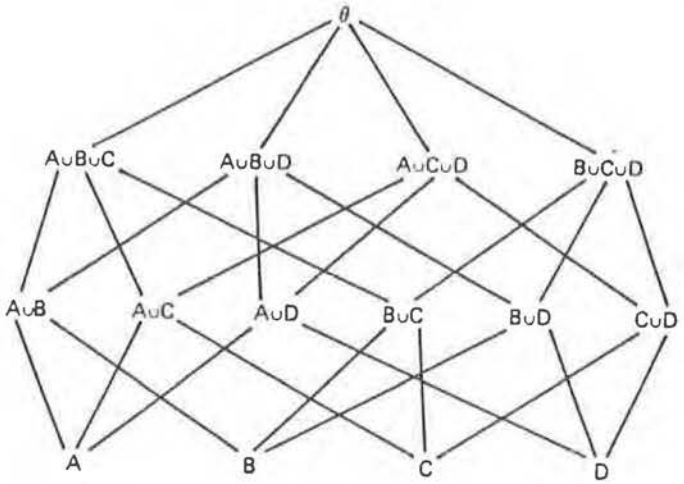


Figure 3: Lattice depicting frame of discernment $\Theta = \{A, B, C, D\}$

The DS Belief function (Bel) corresponds exactly to the ATMS label, in that the symbolic representation of $Bel(\theta_i)$ is the set of propositions from which θ_i can be proven. The plausibility function can be derived similarly, using the set of propositions which do not disprove θ_i, i.e. $Pls(\theta_i) = 1 - Bel(\neg\theta_i)$. Pls is a weaker notion of provability than Bel.

We can use a proposition/assumption set lattice to provide a more intuitive notion of this correspondence. Consider the lattice built using assumptions (or propositions) A, B, C, D, as shown in Figure 3. Subset/superset relations are indicated by edges, and sets of propositions by vertices. In this lattice, each vertex logically entails vertices above it to which it is connected by an edge provided that both assumption sets are consistent.

During interpretation construction, the ATMS constructs such a lattice, identifying the maximal consistent assumption sets. The supersets of any inconsistent assumption set are inconsistent as well. Hence the ATMS can be thought of as identifying

the consistent assumption sets to which mass is assigned in DS theory. As shown by Zadeh [47], DS theory cannot combine conflicting evidence (i.e. assumption sets). However, using the ATMS to identify contradictory sets of assumptions as nogoods ensures that this problem never arises in computing DS weights.

ATMS label updating/DS updating correspondence: If the denominator of the DS updating equation (4) is ignored, equation 4 and ATMS label updating are identical. The denominator in equation 4 sums the mass assigned to the non-contradictory assumption sets, i.e. it computes the belief function of $1-\sum Bel$(conflicting assumption sets), which is equal to $1-\sum Bel$(nogood set). Hence the denominator sums the mass assigned to the nogood database.

For this application, the denominator is irrelevant, because we want to use the DS mass functions to rank interpretations, and not to explicitly compute the belief assigned to each interpretation. Ignoring the denominator provides the interpretation ordering we want, as each interpretation would just be multiplied by the denominator.

This semantic correspondence means that the ATMS can perform most of the computational work necessary to compute DS mass functions. We discuss the additional complexity of computing DS mass functions in Section 4.1.

3.2 Computing Dempster Shafer Belief Functions

The ATMS can be extended with DS theory quite simply. To each ATMS assumption A_i assign a DS mass function $\varrho(A_i)$. Then use the ATMS to symbolically compute the labels for all the literals. Similar to the ATMS's removal of contradictory sets (nogoods) from all contexts as an essential part of its processing, DS theory can assign belief only to non-contradictory subsets, i.e. it "removes" mass assigned to contradictory subsets. That is, the numerator of equation 4 assumes non-conflicting subsets. Hence, θ is not provable for subsets which have a non-null intersection with contradictory subsets. In the ATMS implementation, nogoods must be explicitly accounted for in computing DS Belief functions. This is done by assigning belief in the numerator of equation 4 only to subsets with null intersections with nogoods (i.e. conditioning on the consistent sets), and using a normalisating function, the denominator of equation 4, to renormalise all Belief assignments given that no mass is assigned to nogoods. Hence, the Belief assigned to any proposition (literal) is given by:

The un-normalised DS mass function for any literal x can be computed from its label as follows. We discuss these computations more fully in Section 4. To provide an intuitive understanding, we give the simplest algorithm, described in terms of ATMS labels and Belief functions.

1. Compute a Boolean expression from the label: $\mathcal{L} = \{L_1 \vee L_2 \vee L_3\}$, where each $L_i = \wedge_k A_k$ for the set of k assumptions.

2. Account for nogoods

$$\begin{aligned} Bel(x) &= Bel[\mathcal{L}(x) \mid \neg nogood] \\ &= \frac{Bel[\mathcal{L}(x) \cap \neg nogood]}{Bel[\neg nogood]} \\ &= \frac{Bel[\mathcal{L}(x)] - Bel[\mathcal{L}(x) \cap nogood]}{1 - Bel[nogood]}. \end{aligned} \tag{5}$$

   Alternatively, if we use the relation

$$\varrho(A \cap B) = \varrho(A) + \varrho(B) - \varrho(A \cup B),$$

   we obtain

$$Bel(x \mid \neg nogood) = \frac{Bel[\mathcal{L}(x) \cup nogood] - Bel[\mathcal{L}(x)]}{1 - Bel[nogood]}, \tag{6}$$

   which is Dempster's Rule of Conditioning.

3. Convert the Boolean expression 5 or 6 into a form such that all the L_i are mutually exclusive or independent, which we call MEI (mutually exclusive/independent) form.

4. Convert the Boolean expression into an expression representing a Dempster-Shafer mass function, using the intersection formula

$$\varrho(A_1 \wedge A_2) = \varrho(A_1)\varrho(A_2) \tag{7}$$

   and union formula

$$\varrho(A_1 \vee A_2) = \varrho(A_1) + \varrho(A_2) - \varrho(A_1)\varrho(A_2). \tag{8}$$

5. Substitute mass functions for the A_i's to calculate the mass function for x.

Note that in Steps 1 to 4 we manipulate Boolean expressions. We refer to steps 3 and 4 as a Network Reliability computation, and show several more efficient algorithms, as well as some approximation algorithms, for this computation in Section 4.

3.2.1 Examples

Example 1:

Consider a following example without nogoods: the set of clauses is

x_1	
x_6	
$x_1 \wedge A_1 \Rightarrow x_2$	$\overline{x_1} \vee \overline{A_1} \vee x_2$
$x_2 \wedge A_2 \Rightarrow x_3$	$\overline{x_2} \vee \overline{A_2} \vee x_3$
$x_1 \wedge A_3 \Rightarrow x_4$	$\overline{x_1} \vee \overline{A_3} \vee x_4$
$x_4 \wedge A_4 \Rightarrow x_5$	$\overline{x_4} \vee \overline{A_4} \vee x_5$
$x_2 \wedge x_4 \wedge A_5 \Rightarrow x_5$	$\overline{x_2} \vee \overline{x_4} \vee \overline{A_5} \vee x_5$

The masses assigned to the assumptions are:

ASSUMPTION	MASS
A_1	.5
A_2	.7
A_3	.8
A_4	.6
A_5	.9
A_6	.4

The labels the ATMS assigns to the literals are:

LITERAL	LABEL
x_2	$\{A_1\}$
x_3	$\{A_1, A_2\}$
x_4	$\{A_3\}$
x_5	$\{\{A_3, A_4\}, \{A_1, A_3, A_5\}\}$

The computation of the Boolean expressions for (and hence Belief assigned to) these labels is trivial except for the expressions for x_5, which we now show:

$$\begin{aligned} Bel(x_5) &= Bel\Big(\{\{A_3, A_4\}, \{A_1, A_3, A_5\}\}\Big) \\ &= Bel((A_3 \wedge A_4) \vee (A_1 \wedge A_3 \wedge A_5)) \\ &= Bel(A_3 \wedge (A_4 \vee A_1 \wedge A_5)) \\ &= \varrho(A_3) Bel(A_4 \vee A_1 \wedge A_5) \\ &= \varrho(A_3)(\varrho(A_4) + \varrho(A_1)\varrho(A_5) - \varrho(A_1)\varrho(A_4)\varrho(A_5)) \end{aligned}$$

The Belief assigned to the literals is:

LITERAL	BELIEF
x_2	.5
x_3	.35
x_4	.8
x_5	.62

Example 2:

Consider the introduction of a new clause $x_2 \wedge A_6 \Rightarrow \overline{x_4}$, such that $\varrho(A_6) = 0.4$. Since x_4 and $\overline{x_4}$ both are in the same context, a nogood is formed:

$$\begin{aligned} Nogood &= \mathcal{L}(x_6) \wedge \mathcal{L}(\overline{x_6}) \\ &= \{A_3\} \wedge \{A_1, A_6\} \\ &= \{A_1, A_3, A_6\} \end{aligned}$$

The new assignment of Belief to literals is:

LITERAL	BELIEF
"nogood"	.16
x_2	.4
x_3	.28
x_4	.76
x_5	.15

Example 3:

Suppose instead that the clause $x_4 \wedge A_7 \Rightarrow \overline{x_6}$ such that $\varrho(A_7) = 0.4$ is introduced. Similar to Example 2, a nogood $\{A_3, A_7\}$ is created and the revised Belief assignment is given by

LITERAL	BELIEF
"nogood"	.32
x_2	.50
x_3	.35
x_4	.71
x_5	.55

For this application, we are primarily interested in determining the mass assigned to partial solutions (i.e. interpretations), so that only the most likely partial solutions will be explored. This will enable the Problem Solver to reduce the number of assumptions, and thus reduce the size of the problem solved by this extended ATMS. This can be done, for example, by (1) deferring the exploration of contexts of low likelihood to produce complete solutions, and (2) "ignoring" certain assumptions which are not members of the current contexts[8], effectively reducing the size of the search space.

[8] One method is to garbage collect these assumptions along with other useless assumptions, as described in [13].

4 Implementation Issues—Network Reliability Computation

4.1 Complexity Considerations

Up to the identification of the label set and of the set of interpretations (i.e. the minimal polynomial $\mathcal{F}$), the computational efficiency of our method relies on the ATMS being able to avoid significant regions of the search space. Provan, in [31] and [33] shows that, in the worst-case, the problem of label generation is of exponential complexity, and the interpretation construction problem is NP-hard.

On top of this, computing the DS mass function either from the ATMS label set Ξ^* or from the minimal polynomial $\mathcal{F}$ is of complexity exponential in the number of literals in the Boolean polynomial F. This problem, referred to as the Network Reliability problem, has been studied extensively.

From this summary of the complexity of the sub-problems of computing DS Belief functions using the ATMS, we see that it is necessary to solve two (and possibly three) problems of exponential worst-case complexity, label generation, interpretation construction (which is not necessary) and network reliability. Hence, using the ATMS to compute DS Belief functions encounters an exponential worst-case complexity similar to that which has prevented implementation of the full DS theory up until now. If one is using the ATMS already, DS Belief functions may be computed or approximated efficiently, given an efficient network reliability algorithm.

The difficult part of the network reliability problem is transforming a DNF Boolean function F into mutually exclusive/independent (or series/parallel) form, or some equivalent form from which the DS weights can be simply computed. The actual assignment of DS mass functions to assumptions and computation of mass functions once F is in the appropriate form is trivial.

Since we are interested only in relative ranking of interpretations, we have designed the system to efficiently compute approximate solutions using heuristics which maintain the relative ranking (with respect to DS masses) of the interpretations. Consequently, by determining *approximate* rather than exact solutions to the network reliability problem, the system avoids the intractability associated with computing the *exact* solution.

We now briefly review the Network Reliability problem, and show the approximation we use.

4.2 Network Reliability

In describing the Network Reliability problem, we need to introduce some graph-theoretic notation, such as that defined in any text on graph theory, e.g. [46]. This is because many network reliability solution methods are based on exploiting the properties of the

underlying graph. In making these definitions we show the equivalence of the graph-theoretic and logical descriptions of the problem.

We can consider the Boolean function F as defining a graph $\mathcal{G}(V, E)$ composed of vertices V and edges E. More precisely, a Boolean literal x_i corresponds to an edge E_i, and a logical connective $(\vee, \wedge)$ corresponds to a vertex that joins two or more edges between the corresonding components as follows: a $\wedge$ connecting two literals (or clauses) corresponds to an edge connecting two vertices (or vertex sets) in series, and a $\vee$ connecting two literals (or clauses) corresponds to an edge connecting two vertices (or vertex sets) in parallel. The direction of the edges corresponds to the direction of implication for the clauses.

Further, we assume each edge is associated with a statistically independent random variable with only two possible states, functioning or not functioning. An *event* is a state assignment to literals or an instantiation of variables, such as x_1 to x_{n-1} functioning and x_n not functioning. There are 2^n possible events. We assign to each edge a $[0, 1]$ weight, $\varrho : E \rightarrow [0, 1]$, which is the probability that the edge functions properly. This probability corresponds to the weight assigned to the ATMS assumption that the edge represents. We will refer to a *system* as a set of clauses which can be thought of as a graph or a Boolean polynomial.

A *path* consists of a set of connected edges. We call $\mathcal{E}$ a path between vertices s and t in the event that all edges in the path are functioning. A *minimal path* is a path the deletion of any edge of which renders the path disconnected. A *circuit* is a closed path. A *subgraph* $\mathcal{G}'(V', E')$ of $\mathcal{G}(V, E)$ is a graph such that $V' \subseteq V$ and $E' \subseteq E$. A graph is connected if it has at least one path between every pair of vertices. A cutset of a graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ the removal of any edge (or vertex) of which renders $\mathcal{G}$ disconnected. In the following we refer only to edge cutsets, as we are concerned only with models in which edges fail.[9] A *tree* T is a subgraph with no circuits. A *max-tree* T^* is a tree consisting of $\mid V \mid -1$ edges. The *order* of a vertex is the number of edges incident on it. The order of a graph is the order of a vertex if all vertices have the same order, of the average of the orders of all vertices. A *p-graph* is a subgraph $\mathcal{G}'(V', E')$ of $\mathcal{G}(V, E)$ such that each $e \in E'$ is on a path from a source vertex s to a terminal vertex t. For example, the simplest p-graph is an $s - t$ path. An *acyclic p-graph* is a p-graph with no cycles.

The reliability $R(s, t)$ is the probability that an $s - t$ path exists, and we define this version of the reliability problem as follows:

Reliability Problem: Given a graph $\mathcal{G}(V, E)$ and a probability assignment ϱ to E, compute the probability that a functioning path exists between two distinguished vertices s and t.

We note that there are many other graph-theoretic definitions of reliability, such as:

- the reliability between a distinguished vertex s and a set $K \subseteq V$, $s \notin K$, of vertices,

[9] Reliability models in which vertices, or both edges and vertices fail have also been explored.

Table 1: Correspondence between Boolean and Network Reliability forms

Boolean Form	Network Reliability Form
x_i	$(1 - \varrho(x_i))$
$\overline{x_i}$	$\varrho(x_i)$
$\wedge$	arithmetic product
$\vee$	arithmetic sum

- the reliability between all pairs of vertices.

The reliability problem defined here is NP-hard [4]. The complexity of other definitions of the reliability problem has been examined in [44], [5], [36] and [4]. In general, all reliability problems, except for a few special cases, are NP-hard in the worst-case.

4.3 Relation of the ATMS to Graph Theory/Network Reliability

In this section we outline the relationship between the problem we compute with the ATMS and the network reliability problem. We use the labels for the literals (as generated by the ATMS) to compute the DS Belief function, which is a reliability measure.

We have already shown how a Boolean polynomial F defines a graph. Now, we outline the graph-theoretic equivalent of the labels. But instead of the label, we will use a closely-related type of clause, the prime implicate. A label is just another means of representing a prime implicate Π, and Π is the representation necessary to compute the DS Belief function (or reliability). Using the notion of prime implicate defined in Section 2.2, it can be shown that the set of prime implicates for F, $\Pi(X)$, defines a set of paths through $\mathcal{G}$. If F is expressed in DNF, the set of prime implicants for X is equivalent to the cut sets for the corresponding graph.

The prime implicates contained in the minimal polynomial $\mathcal{F}$ are non-unique and hence are equivalent to a non-unique minimal path set. The use of a stronger minimisation criterion, that of s-coherent irredundancy, can produce a unique minimal polynomial. An *s-coherent* polynomial is a polynomial consisting of prime implicates containing negated literals only.[10] An s-coherent irredundant polynomial $\mathcal{F}_{sc}$ is unique and hence defines the unique minimal path set. If $\mathcal{F}_{sc}$ is in DNF, it defines the unique minimal cut sets. Hence $\mathcal{F}_{sc}$ defines a graph consisting of only and all the minimal paths, and is the minimal representation of the graph for the purposes of computing the network reliability. That is, $\mathcal{F}_{sc}$ and the equivalent system reliability formula are termwise identical; the operations necessary to convert $\mathcal{F}_{sc}$ (in CNF) to a network reliability formula are given in Table 1.

[10] See [9] for a full definition.

Similar to there being a number of prime implicates exponential in the number of literals or clauses, the number of $s-t$ path-/cutsets is an exponential function of $\mid V \mid$ and $\mid E \mid$.

We use this subset of correspondences to illustrate how the graphical and logical methods are just two different ways of looking at the same problem. We now examine methods of obtaining a Boolean polynomial equivalent to $\mathcal{F}_{sc}$.

4.4 Methods of Solving Network Reliability Problems

The Network Reliability problem has been studied extensively in the literature. Several methods have been developed for computing network reliability. The computational approaches fall into three categories of techniques:

1. Path/cutset enumeration methods
2. Pivotal factoring/decomposition
3. Topological decomposition.

Each approach simply ensures that conditional dependencies are taken care of in the reliability computation. For example, the path/cutset enumeration approach ensures that no pair of paths/cutsets have an overlap (i.e. must not share a sub-path), which would make the pair conditionally dependent. The pivotal decomposition method ensures that no event (in this case an edge) is double-counted by expanding the reliability fromula $R(s,t)$ based on on two disjoint events, an edge working $R(s,t \mid E_j)$ and it not working $R(s,t \mid \overline{E_j})$, as shown in Equation 13. We discuss each of these techniques in turn.

4.4.1 Path/Cutset Enumeration

The path/cutset enumeration methods begin with the (minimal) set of paths/cutsets, and expand them so that they are conditionally independent. The two expansion methods used are:

1. Inclusion/exclusion
2. Sums of Disjoint products

We note that the input to algorithms based on these methods, minimal paths/cutsets, corresponds to the set of prime implicates/implicants.

The reliability of an $s-t$ path $R(s,t)$ is given by the sum of the probabilities that the r paths between s and t, $(\mathcal{E}_1, ..., \mathcal{E}_r)$ exist:

$$R(s,t) \;=\; \varrho(\bigcup_{k=1}^{r} \mathcal{E}_k).$$

Enumerating the minimal cutsets of a graph is equivalent to enumerating the minimal paths, by Menger's Theorem (see [46]). We note that for any graph $\mathcal{G}(V,E)$, there are $2^{|E|-|V|+2}$ possible paths between any nonadjacent pair of nodes; for large graphs, the number of paths or cutsets is very large.

The cutset-based reliability of an $s-t$ path $R(s,t)$ is given by

$$R(s,t) \;=\; 1 - P(\bigcup_{i=1}^{N} C_{s,t}^{i}), \tag{9}$$

where $C_{s,t}^{i}$ is the event that all edges fail in the i-th prime cutset and N is the total number of prime cutsets with respect to nodes s and t. As in the computation of $R(s,t)$ by path enumeration, each cutset must be conditionally independent. For a graph $\mathcal{G}(V,E)$, the order of the number of cutsets is $2^{|V|-2}$, as compared to $2^{|E|-|V|+2}$ paths. For graphs with average degree ≥ 4, $\mid E \mid > 2 \mid V \mid$ and $2^{|E|-|V|+2} > 2^{|V|-2}$, i.e. there are more paths than cutsets. Hence, for such graphs enumerating the cutsets is more efficient.

Inclusion/exclusion (IE)

This method is based on the following simple expansion of parallel and series links:

- parallel links are computed using

$$\varrho(A_1 \wedge A_2) = \varrho(A_1)\varrho(A_2),$$

- and series links using

$$\varrho(A_1 \vee A_2) = \varrho(A_1) + \varrho(A_2) - \varrho(A_1)\varrho(A_2).$$

Given a path set $(\mathcal{E}_1, ..., \mathcal{E}_r)$, the reliability is given by

$$\begin{aligned} R(\mathcal{G}) \;=\; & \sum_{i=1}^{r} \varrho(\mathcal{E}_i) - \sum_{i=1}^{r}\sum_{j<i} \varrho(\mathcal{E}_i\mathcal{E}_j) \;+\; \cdots \\ & \cdots \;+\; (-1)^{r-1}\varrho(\mathcal{E}_1\mathcal{E}_2\cdots\mathcal{E}_r). \end{aligned} \tag{10}$$

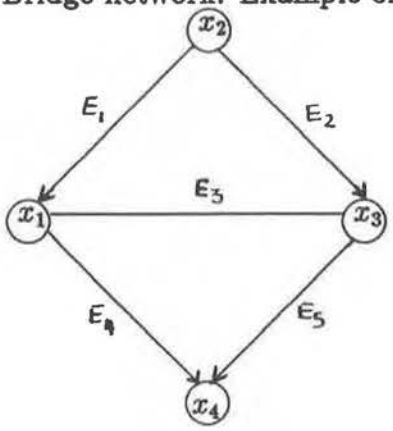
Here $R(\mathcal{G})$ means the general network reliability given a any network measure, i.e. not necessarily a measure such as an $s-t$ path. We note that this way the technique demonstrated in Section 3.2.1. As shown by equation 10, the terms alternate in sign, with the terms with $-$ signs being the double-counted terms. An example of this enumeration technique is [22].

Sum of Disjoint Products (SDP)

This method is based on expanding all parallel paths using the following formula:

$$\varrho(\mathcal{E}_1 \vee \mathcal{E}_2) = \varrho(\mathcal{E}_1) + \varrho(\overline{\mathcal{E}_1} \wedge \mathcal{E}_2). \tag{11}$$

Figure 4: Bridge network: Example of a non-MEI formula



Thus, for a system with s paths, we obtain

$$R = \varrho(\mathcal{E}_1) + \varrho(\overline{\mathcal{E}_1}\mathcal{E}_2) + \ldots + \varrho(\overline{\mathcal{E}_1\mathcal{E}_2}\cdots\overline{\mathcal{E}_{r-1}}\mathcal{E}_r). \tag{12}$$

This method generates s terms for s path sets, but takes exponential time to generate each term in the worst case. We note that an SDP reliability formula contains fewer terms than the equivalent IE formula for all but the smallest systems, and for large systems is a factor of 10 smaller.

This technique was first explored by Fratta and Montanari [17], and then improved upon by Grnarov et. al [20] and Abraham [1]. The Abraham method has since been improved by Locks [24] and Beichelt and Spross [7].

4.4.2 Pivotal Decomposition/Factoring

This method can be used for any graph (formula), and is especially useful for graphs (formulae) which can not be reduced to a set of series/parallel (MEI) paths, such as that representing the bridge network shown in Figure 4. This method is based on the factoring theorem, which "factors out" edges in a graph by conditioning on such edges. Thus, it conditions on the functioning of some edge E_j such that

$$R(s,t) = p_e R(s,t \mid E_j) + (1-p_e)R(s,t \mid \overline{E_j}), \tag{13}$$

where p_e is the probability that edge E_j is functioning. If we use this method to create an MEI graph from a non-MEI graph, we choose an edge which prevents enumeration of MEI paths. Edge E_3 in Figure 4 corresponds to the edge which reduces the graph to MEI form using the factoring theorem.

We note that the input to this method is either a graph or Boolean formula, i.e. it does not need the set of minimal paths or cutsets. An examples of this technique is [28].

4.4.3 Topological Decomposition

Topological decomposition is a method introduced by Satyanarayana and Prabhakar [39] which reduces a graph $\mathcal{G}$ to its p-acyclic subgraphs. The system reliability can be computed directly from the set of p-acyclic subgraphs, due to the direct correspondence between reliability formula and p-acyclic subgraphs. This method is related to the Inclusion/Exclusion method, but it generates only the noncanceling terms of equation 10. Hence, it is more efficient than IE with respect to the number of terms generated as it generates only half the terms of equation 10. This algorithm is based on domination theory (see [2], [45]), and an explanation of it is beyond the scope of this paper.

4.4.4 Applying Exact Methods

Given that the ATMS has already computed the (minimal) prime implicates, we use network reliability methods based on path/cutset enumeration or pivotal factoring, since their input can be a set of prime implicates. The path/cutset enumeration methods can use either the prime implicates Π (from which the ATMS labels are derived) or the minimal prime implicates Π^* (derived from the minimal polynomial $\mathcal{F}$). We now briefly discuss the tradeoff involved in using Π or Π^*.

If Π^* is used, the ATMS must convert Π to Π^*, which is an NP-hard problem. In generating the reliability formula $R(\mathcal{G})$, there is an exponential increase in the number of terms. In this case $\mid R(\mathcal{G}) \mid = O(2^{\mid \Pi^* \mid})$.

If Π is used, the ATMS does not need to convert Π to Π^*. However, typically $\mid \Pi \mid \gg \mid \Pi^* \mid$, and the resultant reliability formula will contain significantly more terms, and be more computationally expensive to compute. We have yet to determine which approach is more efficient in general.

In general, of the Path/Cutset methods, SDP algorithms are more efficient than IE algorithms because fewer terms are generated. Locks [25] has shown that the revised Abraham method is the best SDP algorithm known to date. No detailed comparisons of algorithms based on Path/Cutset and Pivotal Decomposition/Factoring methods exist, to the author's knowledge.

We also advocate using restrictions wherever they are applicable to reduce the computational complexity. For example, in the case of hierarchical evidence, the algorithm of Shafer and Logan [41] can be used, giving a complexity only linear in the number of elements of Θ, instead of exponential for the general case of non-hierarchical evidence.

4.4.5 Approximation Methods

To avoid the exponential complexity associated with computing exact reliability formulae, various approximation methods have been proposed. We outline a few approximation methods based on path/cutset enumeration methods, since these are the methods which

most concern us, given that these methods can accept prime implicate input (i.e. use the ATMS labels).

One such approximation method is based on tree-enumeration technique of Fu [18]. Fu set $Prob(T_i^*)$ to be the probability that tree T_i^* exists, and for edge set $\{E_1, ..., E_r\}$,

$$Prob(T_i^*) = \prod_{j \text{ such that } E_j \in T_i^*} \varrho_j.$$

Hence the reliability of graph $\mathcal{G}$ is approximated by

$$Prob(\mathcal{G}) \simeq \sum_{i=1}^{|T|} Prob(T_i^*).$$

For the trees we substitute the minimal paths (prime implicates) $\mathcal{E}_1, ..., \mathcal{E}_r$ contained in the minimal polynomial $\mathcal{F}$ to obtain an upper bound $U - Prob(\mathcal{G})$ for the reliability expression $Prob(\mathcal{G})$ as given by Equation 10. This method ignores inter-path conditional dependencies, and its inaccuracy is dependent on the degree of such dependencies. Using this method, it can be shown that

$$U - Prob(\mathcal{G}) \simeq \sum_{i=1}^{r} Prob(\mathcal{E}_i) = det(MRM'), \tag{14}$$

where M is the node incidence matrix of $\mathcal{F}$, R is the diagonal edge reliability matrix, and M' is the transpose of M. In many cases both M and R are sparse, and so sparse matrix methods can be used to speed computations further.

Similarly, a lower bound can be obtained as

$$L - Prob(\mathcal{G}) = \sum_{i=1}^{r} Prob(\mathcal{E}_i) - \sum_{i=1}^{r} \sum_{j<i} Prob(\mathcal{E}_i \mathcal{E}_j). \tag{15}$$

These bounds can be improved by including more terms from the expansion given by Equation 10, as (14) corresponds to the first RHS term of (10). The extent of the expansion determines the accuracy of the approximation. Using this IE expansion method, Fong and Buzacott [16] describe algorithms for improving the bounds given by (14) and (15) through similar partial expansions of (10).

Alternatively, the set of prime implicates in conjunction with the prime implicants can be used to obtain upper and lower bounds on $R(\mathcal{G})$, as noted by Chu and Apostolakis [9]. If we order the set of prime implicates as $\Pi = \{\Pi_1, \Pi_2, ..., \Pi_u\}$, and the prime implicants as $\Gamma = \{\Gamma_1, \Gamma_2, ..., \Gamma_v\}$, $R(\mathcal{G})$ can be bounded by:

$$\max_{1 \le j \le v} \{ \bigcap_{i \in \Gamma_j} \varrho(x_i)(1 - \varrho(x_i))^{1-e_i} \} \le R(\mathcal{G}) \le \min_{1 \le k \le u} \{ \bigcup_{l \in \Pi_k} \varrho(x_l)(1 - \varrho(x_l))^{1-e_l} \}, \tag{16}$$

where $\cap$ indicates a product and $\cup$ indicates a sum. The drawback of this method for this ATMS application is that it entails computation of Γ, which is non-trivial.

Chu and Apostolakis also describe several other bounds on $R(\mathcal{G})$ based on Π and Γ. Provan [35] and Locks [26] also detail several reliability bounds. We note that the evaluation of the bounds described here (e.g. in terms of accuracy and increased efficiency over exact values), and the determination of better bounds is still an open research issue.

In general, the choice of approximation technique is dependent on how close the approximation is required to be to the exact value, and to some extent on problem characteristics. At present, we are using the bounds given by (14) and (15), primarily because of their computational simplicity and because our present application requires only ranking of solutions, hence making fairly rough estimates of Belief assignments sufficient.

5 Conclusions

We have described an extended ATMS suitable for solving complex diagnostic problems. This TMS incorporates the full DS theory in a semantically clear and efficient manner. Such an extension allows ranking of competing solutions and exploration of only the "most likely" solutions by the Problem Solver.

This extension has moderate computational requirements over and above that of the ATMS. The efficiency of computing DS weights from ATMS labels relies on obtaining approximations to the weights. This is sufficient for the diagnostic application we propose, as all we require is a relative ordering of the interpretations. Exact DS weights can be computed from ATMS labels, but at significantly greater computational expense.

Acknowledgements: Johan de Kleer has provided helpful comments. Judea Pearl's discussion of the semantics of the ATMS and DS theory have helped refine my understanding of their correspondence, as have discussions of network reliability algorithms.

References

[1] J.A. Abraham. An Improved Method for Network Reliability. *IEEE Trans. Reliability*, R-28:58–61, 1979.

[2] A. Agrawal and R.E. Barlow. A Survey of Network Reliability and Domination Theory. *Operations Research*, 32(3):478–492, 1984.

[3] J.F. Baldwin. Evidential Support Logic Programming. *Fuzzy Sets and Systems*, 24:1–26, 1985.

[4] M.O. Ball. Computational Complexity of Network Reliability Analysis: An Overview. *IEEE Trans. Reliability*, R-35:275–285, 1986.

[5] M.O. Ball. The Complexity of Network Reliability Computations. *Networks*, 10:153–165, 1977.

[6] J.A. Barnett. Computational Methods for a Mathematical Theory of Evidence. In *International Joint Conference on Artificial Intelligence*, pages 868–875, 1981.

[7] F. Beichelt and L. Spross. An Improved Abraham-Method for Generating Disjoint Sums. *IEEE Trans. Reliability*, R-36:70–74, 1987.

[8] C. Chang and R.C. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, NY, 1973.

[9] T.L. Chu and G. Apostolakis. Methods of Probabilistic Analysis of Noncoherent Fault Trees. *IEEE Trans. Reliability*, R-29:354–360, 1980.

[10] B.D. D'Ambrosio. Combining Symbolic and Numeric Approaches to Uncertainty Management. In *AAAI Uncertainty in Artificial Intelligence Workshop*, pages 386–393, Morgan Kaufmann, 1987.

[11] R. Davis. Diagnosis based on structure and function. In *Proceedings of the American Association for Artificial Intelligence*, pages 137–142, 1982.

[12] R. Davis. Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence Journal*, 24:347–410, 1984.

[13] J. de Kleer. An Assumption-based TMS. *Artificial Intelligence Journal*, 28:127–162, 1986.

[14] J. de Kleer and B. Williams. Diagnosing multiple faults. *Artificial Intelligence Journal*, 32:97–130, 1987.

[15] J. Doyle. A Truth Maintenance System. *Artificial Intelligence Journal*, 12:231–272, 1979.

[16] C.C. Fong and J.A. Buzacott. Improved Bounds for System-Failure Probability. *IEEE Trans. Reliability*, R-36:454–458, 1987.

[17] L. Fratta and U. Montanari. Boolean Algebra Method for Computing the Terminal Reliability in a Communications Network. *IEEE Trans. Circuit Theory*, CT-20:203–211, 1973.

[18] Y. Fu. Applications of Topological Methods to Probabilistic Communications Networks. *IEEE Trans. Communication Technology*, 13(3):301–307, 1965.

[19] J. Gordon and E. Shortliffe. A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space. *Artificial Intelligence Journal*, 26:323–357, 1985.

[20] A. Grnarov, L. Kleinrock, and M. Gerla. A New Algorithm for Network Reliability Computation. In *Proc. 1979 Computer Networking Symposium*, pages 17–20, 1979.

[21] L.J Holtzblatt. Diagnosing Multiple Failures Using Knowledge of Component State. In *Proc. IEEE Conf. on AI Applications*, pages 139–143, 1988.

[22] Y. Kim, K. Case, and P. Ghare. A Method for Computing Complex System Reliability. *IEEE Trans. Reliability*, R-21(4):215–219, 1972.

[23] K. Blackmond Laskey and P.E. Lehner. Belief Maintenance: An Integrated Approach to Uncertainty Management. In *Proceedings of the American Association for Artificial Intelligence*, page , 1988.

[24] M.O. Locks. A Minimizing Algorithm for Sum of Disjoint Products. *IEEE Trans. Reliability*, R-36:445–453, 1987.

[25] M.O. Locks. Recursive Disjoint Products: A Review of Three Algorithms. *IEEE Trans. Reliability*, R-31:33–35, 1982.

[26] M.O. Locks. Recursive Disjoint Products, Inclusion-Exclusion, and Min-Cut Approximations. *IEEE Trans. Reliability*, R-29:368–371, 1980.

[27] D. McAllester. *Reasoning Utility Package User's Manual.* Technical Report AIM-667, MIT AI Laboratory, 1982.

[28] K.B. Misra. An Algorithm for the Reliability Evaluation of Redundant Netowrks. *IEEE Trans. Reliability*, R-19:146–151, 1970.

[29] J. Pearl. *Non-Bayesian Formalisms for Managing Uncertainty.* Technical Report R-106, UCLA Department of Computer Science, 1988.

[30] H. Prade. A Computational Approach to Approximate and Plausible Reasoning with Applications to Expert Systems. *IEEE Trans. PAMI*, 7:260–283, 1985.

[31] G. Provan. A Complexity Analysis of Assumption-Based Truth Maintenance Systems. In B.M. Smith and G. Kelleher, editors, *Proceedings of a Workshop on Truth Maintenance Systems*, Ellis Horwood, 1988.

[32] G. Provan. Complexity Analysis of Multiple-Context TMSs in Scene Representation. In *Proceedings of the American Association for Artificial Intelligence*, pages 173–177, 1987.

[33] G. Provan. *Computational Complexity of Truth Maintenance Systems.* Technical Report 88-11, University of British Columbia, Department of Computer Science, 1988.

[34] G. Provan. Solving Diagnostic Problems Using Extended Truth Maintenance Systems. In *European Conference on Artificial Intelligence*, pages 547–552, 1988.

[35] J.S. Provan. Bounds on the Reliability of Networks. *IEEE Trans. Reliability*, R-35:260–268, 1986.

[36] J.S. Provan and M.O. Ball. The Complexity of Counting Cuts and Computing the the Probability the a Graph is Connected. *SIAM J. Computing*, 12:777–788, 1983.

[37] J.A. Reggia, D.S. Nau, and P.Y. Wang. Diagnostic Expert Systems Based on a Set Covering Model. *Int. J. Man-Machine Studies*, 19:437–460, 1983.

[38] R. Reiter and J. de Kleer. Foundations of Assumption-based Truth Maintenance Systems: Preliminary Report. In *Proceedings of the American Association for Artificial Intelligence*, pages 183–188, 1987.

[39] A. Satyanarayana and A. Prabhakar. New Topological Formula and Rapid Algorithm for Reliability Analysis of Complex Networks. *IEEE Trans. Reliability*, R-27:82–100, 1978.

[40] G. Shafer. *Mathematical Theory of Evidence.* Princeton University Press, 1976.

[41] G. Shafer and R. Logan. Implementing Dempster's Rule for Hierarchical Evidence. *Artificial Intelligence Journal*, 33:271–298, 1987.

[42] E.H. Shortliffe and B.G. Buchanan. A Model of Inexact Reasoning in Medicine. In *Mathematical Biosiences*, pages 351–379, 1975.

[43] T. Thompson. Parallel Formulation of Evidential-Reasoning Theories. In *International Joint Conference on Artificial Intelligence*, pages 321–327, 1985.

[44] L. Valiant. The Complexity of Enumeration and Reliability Problems. *Siam J. Computing*, 410–421, 1979.

[45] R.R. Willie. A Theorem Concerning Cyclic Directed Graphs with Applications to Network Reliability. *Networks*, 10:71–78, 1980.

[46] R.J. Wilson. *Introduction to Graph Theory.* Longman, second edition, 1979.

[47] L. Zadeh. Review of Shafer's Mathematical Theory of Evidence. *Artificial Intelligence Magazine*, 5:81–83, 1984.