

AN ANALYSIS OF ATMS-BASED TECHNIQUES FOR COMPUTING DEMPSTER-SHAFER BELIEF FUNCTIONS

Gregory M. Provan*

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

Abstract

This paper analyzes the theoretical underpinnings of recent proposals for computing Dempster-Shafer Belief functions from ATMS labels. Such proposals are intended to be a means of integrating symbolic and numeric representation methods and of focusing search in the ATMS. This synthesis is formalized using graph theory, thus showing the relationship between graph theory, the logic-theoretic ATMS description and the set-theoretic Dempster Shafer Theory description. The computational complexity of calculating Belief functions from ATMS labels using algorithms originally derived to calculate the network reliability of graphs is analyzed. Approximation methods to more efficiently compute Belief functions using this graphical approach are suggested.

1 Introduction

To bridge the gap between the claimed lack of a "logical semantics" in uncertainty calculi and the lack of notions of uncertainty (claimed essential to modeling human reasoning) in logic, several attempts have been made to integrate formal logic with an uncertainty calculus. In this paper the relationships between an uncertainty calculus, Dempster Shafer Theory, and propositional logic are shown.

It has been proposed that Dempster Shafer (DS) Theory rivals Probability Theory in expressive power and effectiveness as a calculus for reasoning under uncertainty. However, because of the computational complexity associated with computing DS Belief functions, only subsets of the full problem domain expressible in DS Theory have been implemented, with the exception of recent Assumption-based TMS (ATMS) implementations. The number of subsets of a set of propositions Θ increases exponentially with $|\Theta|$, and given that the DS normalizing function can sum over all of these subsets, 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 $|2^\Theta| = 2^{2^n}$ in the worst case.

Examples of such restricted implementations include work by Shafer and Logan and by d'Ambrosio. Shafer and Logan [1987] have implemented DS Theory restricted to the case of hierarchical evidence, based on proposals by Barnett [1981] and Gordon and Shortliffe [1985]. D'Ambrosio [1987] has implemented DS theory for the restricted case defined by the Support Logic Programming of Baldwin [1985]. D'Ambrosio attaches a *simplification* of the Dempster-Shafer uncertainty bounds to ATMS labels.

Laskey and Lehner [1988], Provan ([1988b], [1989a]) and Pearl [1988] have independently extended the ATMS with the full DS theory in similar manners. Such an extension represents a synthesis of the symbolic (logic-theoretic) ATMS representation and the numeric (set-theoretic) DS Theory representation. However, there has been no analysis of the underlying theoretical foundations or the complexity of this DS theory implementation.

This paper describes the theoretical and computational issues raised by extending an ATMS to compute DS Belief functions without significantly compromising the semantic clarity or computational properties of either. This extension is motivated by the need to introduce a weighting system into the ATMS, and to improve the poor performance of the ATMS as observed in practice and predicted by the average-case results of Provan [1989b]. Existing ATMSs cannot rank hypotheses, but the incorporation of DS Theory allows hypotheses to be ranked and offers efficiency improvements, such as the ability of the Problem Solver to prune the search space by identifying and focusing search only on highly likely partial solutions. Using a graph theoretic formulation of the ATMS we show that simply weighting the edges of the graph corresponds to a representation from which DS Belief functions can be computed. Such a graph theoretic formulation provides clear intuitions into the properties of symbolic ATMS-based DS algorithms and their relationship to the computation of network reliability [Agrawal and Barlow, 1984].

The complexity of the problems solved when using the ATMS to compute DS Belief functions is also defined.

*The author completed this research with the support of the University of British Columbia Center for Integrated Computer Systems Research, BC Advanced Systems Institute and NSERC grants to A.K. Mackworth.

Such an analysis identifies the sources of intractability in computing DS Belief functions, and explicitly shows why most implementations are of particular restrictions, such as hypertree embeddings [Shenoy and Shafer, 1988], [Shafer and Logan, 1987]. Such theoretical results are supported by empirical evidence (e.g. as pointed out in [Provan, 1986], [Ball, 1986]) which indicates that only moderately-sized problems can be computed efficiently. To counter such intractability, approximation methods for computing the full DS Theory, and not for restrictions of the theory are proposed.

This paper is organized as follows. Section 2 reviews the theoretical basis of DS theory. Section 3 presents three complementary theoretical formulations of the ATMS based on logic, Boolean expression minimization and graph theory. The correspondence between the graph theoretic ATMS formulation and DS theory is shown. Section 4 describes the ATMS-based DS theory algorithm and Section 5 the complexity of the underlying problems. Finally, in Section 6 the conclusions of this study are stated.

2 Dempster-Shafer Theory Review

Many good descriptions of Dempster-Shafer (DS) theory exist, e.g. [Dempster, 1968], [Shafer, 1976]. We assume familiarity with DS theory, 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 a mutually exclusive set of focal propositions Θ . A mass function $\varrho : 2^\Theta \rightarrow [0, 1]$ assigns weights to subsets θ of Θ subject to the following properties: $\varrho(\theta) \in [0, 1]$, $\sum_{\theta \subseteq \Theta} \varrho(\theta) = 1$ and $\varrho(\emptyset) = 0$.

One measure in DS theory which is derived from this mass function is *Belief*, the degree of belief in proposition subsets from which a proposition θ can be proven:

$$Bel(\theta) = \sum_{\varphi \subseteq \theta} \varrho(\varphi). \quad (1)$$

Dempster's Rule of Combination defines an updated mass function for a proposition θ provable in terms of θ_i and θ_j , for all $\theta, \theta_i, \theta_j \subseteq \Theta$, as:

$$\varrho'(\theta) = \frac{\sum_{i,j:\theta_i \cap \theta_j = \theta} \varrho_1(\theta_i) \varrho_2(\theta_j)}{1 - \sum_{i,j:\theta_i \cap \theta_j = \emptyset} \varrho_1(\theta_i) \varrho_2(\theta_j)}. \quad (2)$$

The Belief function ϱ' is also denoted as $Bel_1 \oplus Bel_2$. The numerator assumes independence of propositions. Viewed in set-theoretic terms, this is simply "summing" the mass functions of all sets in which θ is provable.¹ The denominator of Equation 2 is a normalizing term, given that DS Belief is assigned only to non-contradictory subsets.

3 Theoretical ATMS Formalization

The ATMS [de Kleer, 1986] is a database management system which, given a set Σ of propositional clauses,

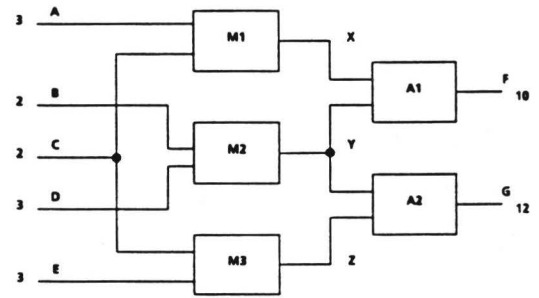
¹DS Theory thus has both underlying set-theoretic and logic-theoretic notions.

computes a label, $\mathcal{L}(x)$, for each database literal x . $\mathcal{L}(x)$ summarizes the "proofs" for x in terms of a Boolean formula consisting of assumptions only. The assumptions, which are denoted by $\mathcal{A} = \{A_1, \dots, A_I\}$, are a distinguished subset of the database literals and are the primitive data representation of the ATMS with which the derivation of all other literals is recorded.

The ATMS records contradictions in terms of a conjunction of assumptions called a *nogood*. By ensuring null intersections of all labels with the set of nogoods, the ATMS maintains a consistent assignment of labels to database literals. The ATMS can incrementally update the database labeling following the introduction of new clauses. It does this by storing the entire label and nogood set to avoid computing them every time they are needed.

A typical problem for which an ATMS is used is circuit diagnosis, such as that done by GDE [de Kleer and Williams, 1987]. The circuit analyzed in [de Kleer and Williams, 1987] consists of multipliers M_1 , M_2 and M_3 and adders A_1 and A_2 , as shown in Figure 1. Assumptions can be: (1) each component is working,

Figure 1: Circuit with faulty components



where $\mathcal{W}_{A_i}$ signifies that adder A_i is functioning correctly, and $\mathcal{W}_{M_i}$ signifies that multiplier M_i is functioning correctly; or (2) input data, e.g. $A = 3$, $B = 2$, etc. In the course of diagnosis, an assumption like $\mathcal{W}_{M_2}$, i.e. " M_2 is working", may be proved incorrect. For the circuit in Figure 1, the output at F is 10 instead of 12, implying that some combination(s) of M_1 , M_2 , M_3 , A_1 and A_2 is(are) faulty. In GDE, the ATMS identifies hypothesized sets of circuit components whose faulty behavior could cause discrepancies between predicted and observed circuit measurements. 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 which explains all the observations.

Given the set of input clauses and assumptions, the ATMS computes what de Kleer and Williams call minimal conflict sets, which are the labels for circuit malfunctions. The two conflict sets are represented logically as $\neg(\mathcal{W}_{A_1} \wedge \mathcal{W}_{M_1} \wedge \mathcal{W}_{M_2})$ and $\neg(\mathcal{W}_{A_1} \wedge \mathcal{W}_{A_2} \wedge \mathcal{W}_{M_1} \wedge \mathcal{W}_{M_3})$. Hence, the malfunctioning of the circuit shown in Figure 1 can be explained by the simultaneous malfunctioning of A_1 , M_1 and M_2 , or

that of A_1, A_2, M_1 and M_3 .

We now present three complementary methods of viewing the problems solved by the ATMS. We first consider the two frameworks traditionally used to formalize the ATMS, namely propositional logic and Boolean algebra. We then show the equivalence of a graphical framework.

3.1 Logic Formulation

The input alphabet (using terminology introduced in [Reiter and de Kleer, 1987]) is represented by a set $x = \{x_1, \dots, x_n\}$ of propositional symbols. A propositional literal is a propositional symbol or its negation. A propositional clause consists of a finite disjunction of propositional literals with no literal repeated, e.g. $\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee \dots \vee \overline{x_k} \vee x$, $k \geq 0$.² $\overline{x_1}, \overline{x_2}, \dots, \overline{x_k}$ are called the *antecedents* of x , $ant(x)$, in keeping with ATMS terminology.

We define an *environment* E to be a set of assumptions and a *context* C to be the set of literals derivable from the assumptions in E given the clauses.

The input consists of a set $\Sigma = \{\Sigma_1, \dots, \Sigma_l\}$ of *Horn clauses*, referred to in [de Kleer, 1986] as justifications. A Horn clause is a clause with at most one unnegated literal.

The ATMS's operation consists of two distinct phases, label manipulation and interpretation construction, which is described 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. A typical label is $\{\{A_1, A_3, A_4\}, \{A_5, A_6\}\}$, showing two environments. In logical terms, the Boolean expression for the label is $(A_1 \wedge A_3 \wedge A_4) \vee (A_5 \wedge A_6)$, or the disjunction of the environments. Each environment in the label for x_i is defined in terms of a minimal support clause ξ^* for x_i , and is given by $\xi^* = \{\bigwedge_{A_j \in A} A_j \mid (\bigvee_{A_j \in A} \overline{A_j}) \text{ is a minimal support clause of } \Sigma\}$. ξ is a *support clause* for x_i with respect to Σ if $\Sigma \models \xi$, $x_i \cup \xi$ does not contain a complementary pair of literals (i.e. both x_j and $\overline{x_j}$), and $\Sigma \models x_i \cup \xi$. ξ^* is a minimal support clause for x_i with respect to Σ if no proper subset of ξ is a support clause for x_i with respect to Σ . A closely related type of clause, a prime implicate,³ is defined as follows: a prime implicate of a set Σ of clauses is a clause π such that (1) $\Sigma \models \pi$, and (2) for no proper subset π' of π does $\Sigma \models \pi'$. The relationship between the label (minimal support clause) for a literal x and a prime implicate for Horn clauses is given as follows [Reiter and de Kleer, 1987]: ξ is a minimal support clause for x with respect to Σ iff there is a prime implicate π for Σ such that $x \in \pi$ and $\xi = \pi - x$. The ATMS thus computes the set $\Pi(\Sigma)$ of prime implicates and assigns as a label for each literal the disjunction of support clauses (restricted to assumptions) for that literal.

²Any clause can be represented as either $x_i \supset x_j$ or $\overline{x_i} \vee x_j$. The latter method is used by convention.

³The 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.

If a new clause $\overline{x_1} \vee \overline{x_2} \vee \dots \vee x$ is added such that the antecedents of x already have labels, or if (x) must be updated due to database updating, label updating must take place. An updated label is assigned to a literal x by taking the conjunction of the labels of the antecedents of x : $\mathcal{L}(x) = \bigwedge_{x_i \in ant(x)} \mathcal{L}(x_i)$. Labels are always represented minimally with respect to set inclusion.

The process of determining solutions, called Interpretation Construction in ATMS terminology, can be described in this logical framework in terms of a CNF to DNF conversion, but is more intuitively described within the Boolean expression framework, in terms of well-known set covering algorithms. We discuss Interpretation Construction in the next section.

3.2 Boolean Algebra Formulation

This formulation uses the same terminology as the logical formulation. The conjunction of the Σ_i 's is called a Boolean expression⁴ F , i.e. $F = \bigwedge_{i=1, \dots, l} \Sigma_i$.

As in the previous formulation, the ATMS label generation algorithm computes the prime implicates for Σ , $\Pi(\Sigma) = \{\pi_1, \dots, \pi_m\}$. The derivation of the prime implicates enables the expression F to be represented in terms of the prime implicates, i.e. $F_\Pi = \bigwedge_i \pi_i$, such that F_Π computes F . A expression F' computes F if $F'(x) = F(x)$ for every instantiation of x . We note that there may be many other expressions F' which also compute F .

The ATMS then derives the *minimal* or *irredundant* expression $\mathcal{F}$ for F . By assigning a unit cost to each support clause ξ_i , the cost of F can be defined as the sum of the costs of the support clauses ξ_i in F .⁵ A minimal expression $\mathcal{F}$ is an expression such that $\mathcal{F}$ computes F and no expression computing F has cost smaller than $\mathcal{F}$.

Interpretation construction derives an irredundant (or minimal) set of prime implicates, where a prime implicate π is redundant if it can be removed from the set Π of prime implicates to form a set Π' such that Π' still computes F . The ATMS uses a set covering algorithm for interpretation construction, but other algorithms may be used, such as the Quine-McCluskey algorithm [McCluskey, 1956], or the hitting set algorithm used by Reiter [1987] for a diagnostic reasoning application. The minimal Boolean expression corresponds in ATMS terminology to an *interpretation*, which de Kleer [1986] defines as the smallest set of assumptions from which all literals in the context are derivable.

For the circuit in Figure 1, a minimal DNF expression $\mathcal{F}$ is $(\overline{W_{A_1}}) \vee (\overline{W_{M_1}}) \vee (\overline{W_{A_2}} \wedge \overline{W_{M_2}}) \vee (\overline{W_{M_2}} \wedge \overline{W_{M_3}})$, which is what de Kleer and Williams call a minimal diagnosis.

3.3 Graph Theoretic Formulation

In describing this formulation, we introduce some graph-theoretic notation, which is used to show the equivalence

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

⁵There are many ways to define such a cost function. For example, the cost of each support clause ξ_i can be assigned as the number of assumptions in ξ_i .

of the graph-theoretic and logic-theoretic descriptions of the problem.

We can consider a Boolean expression F as defining a graph $\mathcal{G}(V, E)$ composed of vertices V and edges E .⁶ 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 corresponding 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.

A *path* consists of a connected sequence of distinct 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 *subgraph* $\mathcal{G}'(V', E')$ of $\mathcal{G}(V, E)$ is a graph such that $V' \subseteq V$ and $E' \subseteq E$. A *connected graph* 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.

Using the correspondence between the Boolean expression and graph theoretic formulations for the ATMS, we outline the graph-theoretic equivalent of the ATMS labels. But instead of the label, we will use the prime implicate π , since a label is just another means of representing π , as shown earlier.

Lemma 1 *The set of prime implicates $\Pi(\Sigma)$ for a CNF Boolean expression F defines a set of paths through the corresponding graph $\mathcal{G}$.*

If F is expressed in DNF, then we obtain

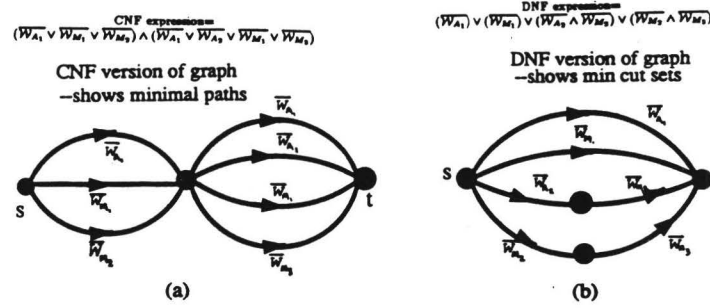
Lemma 2 *The set of prime implicates $\Pi(\Sigma)$ for a DNF Boolean expression F defines the cut sets of the corresponding graph $\mathcal{G}$.*

The graph corresponding to the irredundant CNF expression for the circuit diagnosis example is shown in Figure 2(a). The graph corresponding to the irredundant DNF expression is shown in Figure 2(b) which shows the minimal paths derived from the example. We emphasize that these graphs are interconvertible. By finding the edge cuts containing the minimum number of edges for the graph shown in Figure 2(a), we can obtain a minimal cut representation shown in Figure 2(b). Note the graph theoretic relationships between the diagnostic notions of minimal conflict sets (minimal cut sets) and minimal candidates (minimal paths).

Computing the pathsets or cutsets of a graph $\mathcal{G}$ is one means of computing the network reliability of $\mathcal{G}$, which we now show is analogous to using the ATMS label set to calculate DS Belief function assignments for the corresponding Boolean expression F .

⁶There are many ways of defining a graph from a Boolean expression. We present one method here, and outline other methods in [Provan, 1989a].

Figure 2: Graphs corresponding to circuit diagnosis formulae



4 Correspondence between the ATMS and Dempster-Shafer Belief Functions

4.1 Extension of ATMS Graph

We now show how the simple assignment of weights to the underlying ATMS graph $\mathcal{G}$ enables the computation of DS Belief functions from the very same graph. Assume each edge of $\mathcal{G}$ is associated with a statistically independent random variable with only two possible states, functioning or not functioning. For each DS proposition θ there is an associated ATMS literal x and an antecedent set $\text{ant}(x)$ containing at least one assumption, such as $\overline{x}_3 \vee \overline{A}_2 \vee x$. An *event* is a state assignment to literals, such as $\{x_1, x_2, \dots, x_{n-1}\}$ functioning and x_n not functioning. There are 2^n possible events. Assign to each edge a $[0, 1]$ weight, $\rho : E \rightarrow [0, 1]$, which is the probability that the edge functions properly. Each edge in $\mathcal{G}$ corresponds to an ATMS assumption, and the edge weight corresponds to a weight assigned to the ATMS assumption.

The reliability $R(s, t)$ of a graph $\mathcal{G}$ with respect to two distinguished vertices s and t is defined as the probability that an $s - t$ path exists. The $s - t$ reliability problem is defined as follows:

$R(s, t)$: 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 .

$R(s, t)$ can be computed by summing the probabilities of the disjoint $s - t$ path set. A disjoint path set is a set of paths such that no pair of paths has a common edge. Any pathset, cutset or equivalent Boolean expression for computing $R(s, t)$ must be disjoint.

We summarize the results of [Provan, 1989a] as follows:

Lemma 3 *The label assigned to a literal x is a symbolic (and not necessarily disjoint) representation of the DS Belief assigned to the corresponding DS proposition θ .*

There is an ATMS corollary to DS information pooling (equation 2):

Lemma 4 *The ATMS label updating for a literal x , $\mathcal{L}(x) = \bigwedge_{x_i \in \text{ant}(x)} \mathcal{L}(x_i)$, is a symbolic ATMS analog*

to the DS Belief updating formula for the corresponding proposition θ , $Bel(\theta) = \bigoplus_i Bel_i(\theta)$.

Numeric assignments of Bel can be calculated as given by:

Lemma 5 *The DS Belief assigned to a literal can be computed using an ATMS by converting the weighted ATMS label set to its graphical representation and computing the probability that an $s - t$ path exists in the subgraph formed from the label assigned to the literal.*

Hence calculating DS belief functions for an underlying Boolean expression F is identical to computing the network reliability for the graph corresponding to F .

Moreover, there is an analog in the ATMS to Dempster's rule of Conditioning. Dempster's Rule of Conditioning is as follows:

Lemma 6 *If Bel and Bel' are two combinable Belief functions,⁷ let $Bel(\cdot|\theta_2)$ denote $Bel \oplus Bel'$. Then*

$$Bel(\theta_1|\theta_2) = \frac{Bel(\theta_1 \cup \bar{\theta}_2) - Bel(\bar{\theta}_2)}{1 - Bel(\bar{\theta}_2)} \quad (3)$$

for all $\theta_1 \subset \Theta$.

If we call the set of nogoods Ψ , then the ATMS's symbolic representation of equation 3 is, for all $x \in \mathbf{x}$,

$$Bel(x | \neg\Psi) = \frac{Bel[\mathcal{L}(x) \cup \Psi] - Bel[\Psi]}{1 - Bel[\Psi]}. \quad (4)$$

4.2 Dempster-Shafer Belief Function Algorithm

DS Belief functions can be computed from ATMS labels as follows. To each ATMS assumption A_i assign a DS mass function $q(A_i)$. Then use the ATMS to symbolically compute the labels for all the literals. Similar to the ATMS's removal of 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 because 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: (1) assigning belief in the numerator of equation 2 only to subsets with null intersections with nogoods (i.e. conditioning on the consistent sets); and (2) using a normalization function, the denominator of equation 2, to renormalize all Belief assignments given that no mass is assigned to nogoods. Hence, the Belief assigned to any proposition (literal) can be computed as follows:

1. Compute a Boolean expression from the label, as described in Section 3.1.
2. Account for nogoods, using equation 4, or the equivalent form

$$Bel(x|\neg\Psi) = \frac{Bel[\mathcal{L}(x)] - Bel[\mathcal{L}(x) \cap \Psi]}{1 - Bel[\Psi]}. \quad (5)$$

3. Convert the Boolean expression (4) or (5) into a disjoint form.

⁷cf. [Shafer, 1976], p.67 for a definition of conditions for combinability.

Table 1: Correspondence of Boolean and Network Reliability Forms

Boolean Form	Network Reliability Form
x_i	$q(x_i)$
$\bar{x}_i$	$(1 - q(x_i))$
$\wedge$	arithmetic product
$\vee$	arithmetic sum

4. Substitute mass functions for the A_i 's to calculate the Belief function for x , using Table 1.

Note that in Steps 1 to 3 we manipulate Boolean expressions. We refer to steps 3 and 4 as a Network Reliability computation.

There are many methods for symbolically computing the disjoint form of an arbitrary Boolean expression, some of which are reviewed in [Agrawal and Barlow, 1984]. The methods applicable to this application are those based on pathsets and cutsets, such as the Sum of Disjoint Products and Inclusion-Exclusion methods.

5 Complexity of ATMS-based Dempster-Shafer Belief Function Computation

This extension of the ATMS was originally proposed as a means of improving the efficiency of the ATMS by focusing search only on the portions of the search space in which solutions were most likely to be found. This need arose from the poor performance of the ATMS in practice, and the theoretical evidence ([Provan, 1988a], [Provan, 1989b]) which suggests that (1) for almost all Boolean expressions⁸ the problem of label generation is of complexity exponential in the number of literals; and (2) the interpretation construction problem is NP-hard.

However, the efficiency gains from direct ATMS implementation of DS Theory may not be cost-effective. Using the ATMS to compute DS Belief functions from ATMS labels is intractable in the worst case, a computational barrier similar to that which has previously prevented implementation of the full DS theory.

Lemma 7 *Given the set of ATMS labels for a set $\mathbf{x}$ of literals and an assignment of weights to the ATMS assumption set $\mathcal{A}$, computing the DS Belief assigned to $\mathbf{x}$ either from the ATMS label set Ξ^* or from a minimal expression $\mathcal{F}$ is of complexity exponential in the number of literals in the Boolean expression F , in the worst case.*

If the ATMS label set is used as a starting point, the size of the label set is exponential in the number of literals or clauses [Provan, 1989b], just as the number of $s - t$ pathsets/cutsets is an exponential function of $|V|$ and $|E|$ [Agrawal and Barlow, 1984]. The minimal expression $\mathcal{F}$ may also be used as a starting point, but even

⁸A property is said to hold for almost all the functions of the algebra of logic if the proportion of functions of n variables which do not satisfy this property (among all the functions of n variables) tends to zero when $n \rightarrow \infty$.

though $cost(\mathcal{F}) < cost(F_{\Pi})$, it is unknown whether the savings introduced will be worth computing $\mathcal{F}$ from F_{Π} .

If truth maintenance facilities and numeric assignments of weights to proposition sets are required, this method may be useful, depending on the tradeoff between the cost of computing the DS Belief functions and the search space which can be saved by more focused search. The graphical framework suggests more efficient methods of computing ATMS labels and DS belief functions by exploiting the topology of the underlying graph. Further efficiency improvements may be obtained by using one of the many $R(s, t)$ approximation algorithms applicable to computing DS Belief functions from ATMS labels. In [Provan, 1989a] we describe network reliability *approximation* algorithms which avoid the intractability associated with deriving exact solutions. Such algorithms may remove the necessity to restrict the problem domain to a subset such as that of hierarchical evidence [Shafer and Logan, 1987] due to intractability otherwise.

6 Conclusions

The theoretical basis underlying methods of computing DS Belief functions from ATMS label sets have been discussed. These computations are done using an ATMS which incorporates the full DS theory in a semantically clear and efficient manner, as shown by the graph-theoretic equivalence of the ATMS and DS theory. In addition, computing DS Belief functions from ATMS labels was shown to be intractable in the worst case.

The use of the ATMS to compute DS Belief functions is a promising research area, as it is a means of blending logic with an uncertainty representation in a semantically correct manner. However, we argue that the computational cost of computing *exact* DS Belief functions may be prohibitive for large problems, and approximation methods may have to be used.

Acknowledgements: Discussions with Judea Pearl have helped refine my understanding of the correspondence of the semantics of the ATMS and DS theory, and the use of network reliability algorithms.

References

- [Agrawal and Barlow, 1984] A. Agrawal and R.E. Barlow. A Survey of Network Reliability and Domination Theory. *Operations Research*, 32(3):478–492, 1984.
- [Baldwin, 1985] J.F. Baldwin. Evidential Support Logic Programming. *Fuzzy Sets and Systems*, 24:1–26, 1985.
- [Ball, 1986] M.O. Ball. Computational Complexity of Network Reliability Analysis: An Overview. *IEEE Trans. Reliability*, R-35:275–285, 1986.
- [Barnett, 1981] J.A. Barnett. Computational Methods for a Mathematical Theory of Evidence. In *Proc. IJCAI*:868–875, 1981.
- [D'Ambrosio, 1987] B.D. D'Ambrosio. Combining Symbolic and Numeric Approaches to Uncertainty Management. In *AAAI Uncertainty in AI Workshop*:386–393. Morgan Kaufmann, 1987.
- [de Kleer and Williams, 1987] J. de Kleer and B. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32:97–130, 1987.
- [de Kleer, 1986] J. de Kleer. An Assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.
- [Dempster, 1968] A.P. Dempster. Upper and Lower Probability Bounds. *J. Royal Statistical Society*, 1968.
- [Gordon and Shortliffe, 1985] J. Gordon and E. Shortliffe. A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space. *Artificial Intelligence*, 26:323–357, 1985.
- [Laskey and Lehner, 1988] K. Blackmond Laskey and P.E. Lehner. Belief Maintenance: An Integrated Approach to Uncertainty Management. In *Proc. AAAI*:210–214, 1988.
- [McCluskey, 1956] E.J. McCluskey. Minimization of Boolean Functions. *Bell Syst. Tech. J.*, 35:1417–1444, 1956.
- [Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
- [Provan, 1986] J.S. Provan. Bounds on the Reliability of Networks. *IEEE Trans. Rel.*, R-35:260–268, 1986.
- [Provan, 1988a] G. Provan. A Complexity Analysis of Assumption-Based Truth Maintenance Systems. In B.M. Smith and G. Kelleher, editors, *Reason Maintenance Systems and their Applications*. Ellis Horwood, 1988.
- [Provan, 1988b] G. Provan. Solving Diagnostic Problems Using Extended Truth Maintenance Systems. In *Proc. ECAI*:547–552, 1988.
- [Provan, 1989a] G. Provan. A Logic-Based Analysis of Dempster Shafer Theory. Tech. Report 89-8, University of British Columbia, Department of Computer Science, 1989.
- [Provan, 1989b] G. Provan. The Computational Complexity of Truth Maintenance Systems. Tech. Report 89-9, University of British Columbia, Department of Computer Science, 1989.
- [Reiter and de Kleer, 1987] R. Reiter and J. de Kleer. Foundations of Assumption-based Truth Maintenance Systems. In *Proc. AAAI*:183–188, 1987.
- [Reiter, 1987] R. Reiter. Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–96, 1987.
- [Shafer and Logan, 1987] G. Shafer and R. Logan. Implementing Dempster's Rule for Hierarchical Evidence. *Artificial Intelligence*, 33:271–298, 1987.
- [Shafer, 1976] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
- [Shenoy and Shafer, 1988] P.P. Shenoy and G. Shafer. An Axiomatic Framework for Bayesian and Belief Function Propagation. In *AAAI Workshop on Uncertainty in AI*:307–314. Morgan Kaufmann, 1988.