A Logic-Based Analysis of Dempster Shafer Theory

by

Gregory M. Provan

Technical Report 89-08
December, 1989

Computer Science Department

University of British Columbia

Vancouver, B.C. V6T 1W5 Canada

A LOGIC-BASED ANALYSIS OF DEMPSTER SHAFER THEORY

Gregory M. Provan*
Department of Computer Science
University of British Columbia
Vancouver, BC
Canada V6T 1W5

Abstract

We formulate Dempster Shafer Theory in terms of Propositional Logic, using the implicit notion of provability underlying Dempster Shafer Theory. Dempster Shafer theory can be modeled in terms of propositional logic by the tuple (Σ, ϱ), where Σ is a set of propositional clauses and ϱ is an assignment of measure to each clause $\Sigma_i \in \Sigma$. We show that the disjunction of minimal support clauses for a clause Σ_i with respect to a set Σ of propositional clauses, $\xi(\Sigma_i, \Sigma)$, is a symbolic representation of the Dempster Shafer Belief function for Σ_i. The combination of Belief functions using Dempster's Rule of Combination corresponds to a combination of the corresponding support clauses. The disjointness of the Boolean formulae representing DS Belief functions is shown to be necessary. Methods of computing disjoint formulae using Network Reliability techniques are discussed.

In addition, we explore the computational complexity of deriving Dempster Shafer Belief functions, including that of the logic-based methods which are the focus of this paper. Because of intractability even for moderately-sized problem instances, we propose the use of efficient approximation methods for such computations. Finally, we examine implementations of Dempster Shafer theory, based on domain restrictions of DS theory, hypertree embeddings, and the ATMS.

Keywords: Dempster Shafer Theory, Uncertainty, Logic, Theorem Proving, Assumption-based Truth Maintenance System, Reasoning System.

---

*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 grant A9281 to A.K. Mackworth.

1 INTRODUCTION

It has been claimed that uncertainty calculi lack the semantics of logic and that logic lacks the notions of uncertainty essential to modeling human reasoning, such that both are inadequate for many AI problems. Several attempts have been made to develop new uncertainty calculi, new logics, or to integrate formal logic with an uncertainty calculus to create an adequate knowledge representation language. In this paper we show the relationships between a particular uncertainty calculus, Dempster Shafer Theory, and propositional logic, in an effort to assess the adequacy of Dempster Shafer theory as a knowledge representation tool.

It has been proposed that Dempster Shafer Theory rivals Probability Theory in expressive power and effectiveness as a calculus for reasoning under uncertainty. Probability theory is the best understood uncertainty calculus, both in terms of its philosophical justifications ([Cox, 1946], [Savage, 1954]), and its applicability to AI [Pearl, 1988]. Hence it is the standard by which other uncertainty calculi are judged. There has recently been much study of DS Theory, in terms of its adequacy as an uncertainty calculus ([Pearl, 1988], [Prade, 1985]), and its theoretical underpinnings (especially its relation to probability theory ([Pearl, 1988], [Fagin and Halpern, 1989], [Ruspini, 1987], [Grosof, 1986])).

This paper makes two theoretical contributions and one implementational contribution. The first theoretical contribution is an explicit definition of DS Theory in terms of Propositional Logic, using the implicit notion of provability underlying DS Theory. d'Ambrosio [1987], Laskey and Lehner [1988], Provan ([1988a], [1989]) and Pearl [1988] have recently shown how *Belief* can be defined in terms of provability relations. We formalize and extend those notions.

DS theory was introduced by Dempster [1968] based on statistical notions, but more recently has been described in set theoretic terms [Shafer, 1976]. Shafer [1976] defined a belief measure (Bel) based on the notion of the representation of a *set* of focal propositions Θ in terms of its *subsets*. Given an assignment of mass to each of a mutually exclusive set of focal propositions $\Theta = \{\theta_1, ..., \theta_n\}$, measures of uncertainty, called $Bel(\theta)$, can be assigned to subsets $\theta \in 2^{\Theta}$.

In this paper we describe a logical interpretation of DS theory. We show that DS theory can be characterized in terms of Boolean operations, on top of which a set of constraints (specifically an uncertainty measure) is assigned. The constraints on the propositions are fundamentally Boolean, and the uncertainty measure is secondary. In conducting this analysis, we assume a $[0, 1]$ measure assignment ϱ_i to each element Σ_i of a set $\mathbf{\Sigma}$ of propositional clauses. This is similar to Kong [1986], who viewed each clause as a "joint variable", thus converting a set of clauses into belief network notation.

In addition, we clarify the understanding of Dempster's combination rule by showing its relationship to combining proofs in some minimal fashion. We show that the support set for a clause Σ_i with respect to the set Σ of propositional clauses, $\xi(\Sigma_i, \Sigma)$, when represented in terms of symbols for the ϱ_i's is a symbolic representation of the Dempster Shafer Belief function for Σ_i. We show that the pooling of information, which in DS Theory is represented as $Bel(\theta) = \oplus_i Bel_i$, corresponds to support set combination in our logical formulation. In addition, explicitly computing the numerical value for $Bel(\Sigma_i)$ from its Boolean formula $\xi(\Sigma_i, \Sigma)$,

requires disjointness of the Boolean formula. We show that computing disjoint Boolean formulae is equivalent to evaluating the network reliability of a network defined by Σ or by $\xi(\Sigma_i, \Sigma)$.

This analysis thus describes both how Dempster Shafer (DS) Theory can be assigned a logical semantics and propositional logic can be extended with an uncertainty calculus. This provides insight into how DS uncertainty measures can be assigned to reasoning systems based on logic, such as PROLOG rule-based systems, or truth maintenance systems ([Doyle, 1979], [de Kleer, 1986]).

In the process of analyzing the relationship between DS theory and propositional logic, we clarify the differences between DS theory and probability theory. We show, with respect to probability theory, that DS theory is a complementary (and different) means of assigning uncertainty measures to propositions. It can be defined with respect to notions of logical provability, which probability theory cannot. This underlying notion of provability limits DS theory to situations where a notion of provability is appropriate.[1]

The second theoretical contribution consists of an exploration of the computational complexity of deriving DS Belief functions. We state the complexity of the problem underlying Dempster's Rule of Combination, as well as the complexity of the logic-based algorithms proposed in this paper. Because of intractability, even for moderately-sized problem instances, we propose the use of approximation algorithms. We discuss incorporating some of the techniques for computing the reliability of networks, given the isomorphism between DS Belief function computation and network reliability computation.

The third contribution is an examination of the issues related to implementing DS theory. We briefly examine implementations of DS Theory, including: (1) those based on the traditional subset-relationship approach, (2) those based on hypertree embeddings, and (3) those based on our logical formulation (specifically an implementation within an Assumption-based Truth Maintenance System (ATMS) [de Kleer, 1986]). Because of the computational intractability of these implementations, we describe implementations of restrictions of DS Theory, and we propose efficient DS Belief functions approximation methods.

The remainder of the paper is organised as follows. Section 2 briefly defines several important concepts in DS Theory, including Belief and Plausibility functions and Belief function updating. Section 3 introduces our logical notation. Section 4 defines DS Theory in terms of this notation. Section 5 defines the computational complexity of deriving DS Belief functions. In addition to showing the complexity of Dempster's rule for evidence pooling, we state results for computing the logical functions which correspond to the DS Theory functions. Section 6 examines several implementations of DS Theory, based on domain restrictions of DS theory, hypertree embeddings, and the ATMS. Section 7 discusses related work. Finally, Section 8 summarizes our conclusions.

[1]The three prisoners' dilemma, discussed in [Pearl, 1988], is one example for which DS theory is not applicable.

2 DEMPSTER SHAFER THEORY REVIEW

Many good descriptions of Dempster Shafer theory exist, e.g. [Dempster, 1968], [Shafer, 1976], and secondary sources [Prade, 1985], [Pearl, 1988], [Smets, 1988]. We state a few basic relationships, and refer the reader to the references.

In DS theory, a measure is assigned to elements as well as subsets of a set of focal propositions $\Theta = \{\theta_1, \cdots, \theta_m\}$. The set of focal propositions, called the *frame of discernment*, consists of a set of exhaustive and mutually exclusive propositions. The assignment of measure to each element of the frame of discernment is called a basic probability assignment (bpa). A mass function ϱ assigns weights to supersets of this frame of discernment, $\varrho : 2^{\Theta} \rightarrow [0,1]$, subject to the following properties:

1. $\varrho(\theta) \in [0,1]$ for every $\theta \in \Theta$,
2. $\varrho(\emptyset) = 0$, and
3. $\sum_{\theta \in \Theta} \varrho(\theta) = 1$.

There are several evidence summarizing measures in DS theory which can be derived from this mass function, which include *Belief, Plausibility* and *Commonality.*[2]

Belief is the degree of belief in proposition subsets from which θ can be proven, or the subsets which *necessarily* support θ:

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

Plausibility is the belief in subsets that do not disprove θ, or the subsets which *possibly* support θ:

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

Commonality is the degree of belief which can move freely to all the elements of θ, or the evidence focused on the supersets of θ:

$$Q(\theta) = \sum_{\varphi \supseteq \theta} \varrho(\varphi). \tag{3}$$

Information from multiple sources of evidence over a common set Θ of focal propositions can be pooled using Dempster's Rule of Combination. Thus, for two focal propositions such that $\theta_1 \cap \theta_2 = \theta$, and two bpa's, ϱ_1 and ϱ_2, the combined weight assigned to θ is given by

$$\varrho(\theta) = \frac{\sum_{\theta_1 \cap \theta_2 = \theta} \varrho_1(\theta_1)\varrho_2(\theta_2)}{1 - \sum_{\theta_1 \cap \theta_2 = \emptyset} \varrho_1(\theta_1)\varrho_2(\theta_2)}. \tag{4}$$

[2]These measures can also be characterized without reference to ϱ. See, for example, Theorem 2.1 on p. 39 of [Shafer, 1976].

The denominator of equation 4 ensures that the weight assigned to the empty set is 0. Multiplication of $\varrho_1(\theta_1)$ and $\varrho_2(\theta_2)$ is possible because by assuming the independence of the measures. Equation 4 is also denoted by the combination $\varrho_1 \oplus \varrho_2$, and can be generalized to pooling evidence for an arbitrary number of bpa's, i.e. $\varrho = \bigoplus_{i=1}^{m} \varrho_i$, as given by

$$\varrho(\theta) = \frac{\sum_{\bigcap_i \theta_i = \theta} \varrho_1(\theta_1)\varrho_2(\theta_2)\cdots\varrho_m(\theta_m)}{1 - \sum_{\bigcap_i \theta_i = \emptyset} \varrho_1(\theta_1)\varrho_2(\theta_2)\cdots\varrho_m(\theta_m)}. \tag{5}$$

Note that equation 5 combination is possible only for non-contradictory evidence. Hummel and Landy [1988] show that the problems entailed in pooling contradictory evidence can be avoided by introducing an additional bpa ϱ_0 such that $\varrho_0(\theta) = 0$ for $\theta \neq \emptyset$, and $\varrho_0(\emptyset) = 1$. Updating is possible for all bpa's with the following definition:

$$\varrho_1 \oplus \varrho_2 = \varrho_0 \text{ if } \sum_{\theta_i \cap \theta_j = \emptyset} \varrho_1(\theta_i)\varrho_2(\theta_j) = 1.$$

In a similar manner, Dempster's Rule of Combination defines an updated Belief function for a proposition θ provable in terms of $\theta_1, \cdots \theta_m$ as:

$$Bel(\theta) = \frac{\sum_{\bigcap_i \theta_i = \theta} Bel_1(\theta_1)Bel_2(\theta_2)\cdots Bel_m(\theta_m)}{1 - \sum_{\bigcap_i \theta_i = \emptyset} Bel_1(\theta_1)Bel_2(\theta_2)\cdots Bel_m(\theta_m)}. \tag{6}$$

We denote the combination as

$$Bel(\theta) = (\bigoplus_i Bel_i)(\theta).$$

This equation assumes independence of focal propositions.[3] Viewed in set-theoretic terms, this sums the mass functions of all sets in which θ is provable. In discuss the semantics of Dempster's rule, the denominator of Equation 6 can be ignored, as it is a normalizing term which ensures that the total probability weight will have measure 1. We discuss the use of this normalization term further in Section 4.2.

A Belief function is called Bayesian if each focal element in Θ is a singleton.[4] For this restriction, $Bel(\theta) + Bel(\overline{\theta}) = 1 \quad \forall\theta \in \Theta$, and hence $Pls(\theta) = 1 - Bel(\overline{\theta}) = Bel(\theta)$. In this case a Belief function is an additive measure, and the combination rule (Dempster's rule, equation 4) is equivalent to Bayes' rule with conditional independence of propositions.

In analysing DS theory, representing a DS theory problem in hypergraph notation will prove useful. A hypergraph $\mathcal{H}(V, \mathcal{E})$ consists of a set V of vertices, and a set $\mathcal{E}$ of hyperedges,

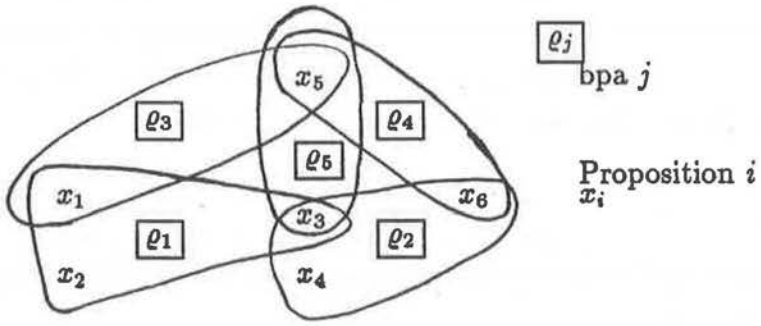
[3] All approaches e.g. Bayes nets, etc. must make independence assumptions of one sort or another for computational tractability.

[4] See [Shafer, 1976], p.44 ff. for a full description of Bayesian Belief functions.

Table 1: Basic probability assignments to a set of focal propositions

Focal Proposition	Measure assignment
x_1, x_2, x_3	ϱ_1
x_3, x_4, x_6	ϱ_2
x_1, x_5	ϱ_3
x_5, x_6	ϱ_4
x_3, x_5	ϱ_5

Figure 1: The hypergraph corresponding to a set of basic probability assignments $\{\varrho_1, \varrho_2, \varrho_3, \varrho_4, \varrho_5\}$.



each of which is a set of vertices. The set V of vertices corresponds in DS theory to the set of atomic propositions; each hyperedge $\mathcal{E}_i$ corresponds to the constraint defined by bpa ϱ_i, such that the vertices in $\mathcal{E}_i$ correspond to the propositions assigned measure by ϱ_i. Each hyperedge emphasizes the notion of a bpa baing a constraint over a set of propositions. The hypergraph notation also helps show the relationship between DS theory and network reliability; the importance of this relationship will be made clear in Section 4.4. As an example of a hypergraph, Figure 1 shows the hypergraph representing the set of atomic propositions $\{x_1, x_2, x_3, x_4, x_5, x_6\}$, and basic probability assignment as given in Table 1. The measure assigned to each hyperedge in $\mathcal{E}$ in shown in the figure enclosed in a box.

We now define what is to be computed using DS theory. Given an assignment of [0,1] weights to a set Θ of focal propositions, the mass assigned to some proposition $\theta \subseteq 2^\Theta$ (or set of propositions), and/or the Belief assigned to some proposition $\theta \subseteq 2^\Theta$ (or set of propositions), is required.

More formally, we define three combination functions: a Belief computation over a single

bpa ϱ, $DSb(\Theta, \varrho, \theta)$, and Mass function and Belief function combinations over multiple bpa's $\vec{\varrho}$, namely $DSM(\Theta, \vec{\varrho}, \theta)$ and $DSB(\Theta, \vec{\varrho}, \theta)$ respectively.

1. *DS Belief ASSIGNMENT* $[DSb(\Theta, \varrho, \theta)]$
   INPUT: A focal proposition set Θ, a weight assignment ϱ over Θ.
   PROBLEM: Compute the Belief assigned to some set $\theta \in 2^\Theta$, i.e. $Bel(\theta) = \oplus_i Bel_i(\theta)$.
2. *DS WEIGHT ASSIGNMENT* $[DSM(\Theta, \vec{\varrho}, \theta)]$
   INPUT: A focal proposition set Θ, a set of weight assignments $\vec{\varrho}$ over Θ.
   PROBLEM: Compute the weight assigned to some set $\theta \in 2^\Theta$, i.e. $\varrho(\theta) = \oplus_i \varrho_i(\theta)$.
3. *DS BELIEF ASSIGNMENT* $[DSB(\Theta, \vec{\varrho}, \theta)]$
   INPUT: A focal proposition set Θ, a set of weight assignments $\vec{\varrho}$ over Θ.
   PROBLEM: Compute the Belief assigned to some set $\theta \in 2^\Theta$, i.e. $Bel(\theta) = \oplus_i Bel_i(\theta)$.

Similar definitions for combination of Plausibility and Commonality functions can be made.

3 PROPOSITIONAL LOGIC REVIEW

We use a propositional language containing a finite set of propositional symbols and the connectives $\vee$, $\wedge$ and $\neg$, defining the connective $\Rightarrow$ in terms of $\vee$ and $\neg$ in the usual way. A propositional *literal* is a propositional symbol or its negation. $\mathbf{x} = \{x_1, \overline{x_1}, ..., x_n\}$ is a set of propositional literals. A *clause* is a finite disjunction of propositional literals, with no repeated literals. $\Sigma = \{\Sigma_1, ..., \Sigma_l\}$ is a set of input clauses. Upper-case, subscripted Σ's represent clauses and lower-case subscripted Σ's represent literals.

For a clause of the form $\overline{x_1} \vee \overline{x_2} \vee x_3 ... \vee x_{m-1} \vee \overline{x_m} \vee x'$, $m \geq 0$, x' is called the *consequent* and $x_1, ..., x_m$ the *antecedents.*[5] A literal is *justified* if it appears as a consequent in a clause. A *Horn clause* is a clause with at most one unnegated literal. For example, a Horn-clause Σ_i can be written as $\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee ... \vee \overline{x_k} \vee x$, $k \geq 0$.

We call $\mathcal{B}$ a Boolean algebra over $\mathbf{x}$. $\mathcal{B}$ is closed under $\neg$, $\vee$ and $\wedge$, with $\Rightarrow$ defined in terms of $\vee$ and $\neg$ in the usual manner. W_i is possible world i, and is the conjunction of a the set of n literals $x_1 \wedge \overline{x_2} \wedge \cdots \wedge x_n$ such that each variable occurs once, either negated or un-negated. $\mathcal{W}$ is the set of possible worlds.

Given this propositional framework, we define two clauses which are derivable from Σ, a prime implicate and a minimal support clause. The latter type of clause provides the notion of provability necessary for characterizing DS theory in logical terms.

A Conjunctive Normal Form (CNF) formula is a formula consisting of the conjunct of disjunctive clauses, e.g. $x_6 \wedge (x_1 \vee x_2) \wedge (\overline{x_3} \vee \overline{x_2} \vee \overline{x_4})$. A Disjunctive Normal Form (DNF)

[5] We often represent a clause not as a disjunction of literals (e.g. $\overline{x_1} \vee x_2$) but as an implication ($x_1 \Rightarrow x_2$). This is done to unambiguously identify the antecedents and consequent.

formula is a formula consisting of the disjunct of conjunctive clauses, e.g. $x_6 \vee (x_1 \wedge x_2) \vee (\overline{x_3} \wedge \overline{x_2} \wedge \overline{x_4})$.

A *prime implicate*[6] of a set Σ of clauses is a clause π (often called $\pi(\Sigma)$ to denote the set Σ of clauses for which this is a prime implicate) such that

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

We denote the set of prime implicates with respect to Σ by Π.

ξ is a *support clause* for x with respect to Σ (often called $\xi(x, \Sigma)$) iff

- $\Sigma \not\models \xi$,
- $x \cup \xi$ does not contain a complementary pair of literals (i.e. both x_j and $\overline{x_j}$), and
- $\Sigma \models x \cup \xi$.

ξ^* is a minimal support for x with respect to Σ iff no proper subset of ξ is a support for x with respect to Σ.

Given a set Σ of clauses, the set of minimal support for Σ can be computed from $\Pi(\Sigma)$, but not vice versa (cf. Theorem 2 of [Reiter and de Kleer, 1987].). In other words, $\Pi(\Sigma) = \xi(\Sigma_i, \Sigma) \vee \Sigma_i$. Hence, the set $\Pi(\Sigma)$ must first be computed, and Ξ^* computed from $\Pi(\Sigma)$. In the case where Σ is a set of Horn clauses, a prime implicate $\pi_k(\Sigma)$ corresponds to the union of a clause and its support clause under the following conditions: $\pi_k = \Sigma_i \cup \xi$, if Σ_i is a unit literal or $\Sigma_i \subseteq \Pi$ (cf. [Reiter and de Kleer, 1987]).

The *set of support* for a literal is the disjunction of the support clauses for that literal, i.e. $\xi(x, \Sigma) = \bigvee_i \xi_i(x, \Sigma)$. We denote the set of supports with respect to Σ by Ξ. The *set of minimal support* for a clause is the disjunction of the minimal support clauses for that clause.

Minimal support clauses provide a means of characterizing simplest explanations (or proofs) (consistent with Σ) for a clause. By definition, $\xi(\Sigma_j, \Sigma)$ is the smallest clause such that $\Sigma \models \neg\xi(\Sigma_j, \Sigma) \Rightarrow \Sigma_j$. A simplest explanation is a conjunction of literals for which no proper sub-conjunct is an explanation. Thus, if $\xi(x, \Sigma) = \overline{x_1} \vee \overline{x_2} \vee \overline{x_3}$, this means that $x_1 \wedge x_2 \wedge x_3$ is a minimal explanation or proof for x. This is formalized in Lemma 1 below.

To model DS theory within this propositional logic framework, a restriction of the notion of minimal support is required. Mark a subset $\mathcal{A} = \{A_1, \cdots, A_l\}$, $\mathcal{A} \subset \mathbf{x}$, of the literals, such that all literals not in $\mathcal{A}$ are derivable from those in $\mathcal{A}$. These marked literals are called assumptions. This is accomplished by ensuring that

1. All assumptions occur as antecedents only; and

[6]The dual to prime implicate (in Boolean algebra) is called a prime implicant. In switching theory prime implicants are used, as the expressions are expressed in DNF, whereas the expression is expressed here in CNF. We use the prime implicate terminology to avoid confusion between the dual representations.

2. All non-assumptions are justified by (a) an assumption, or (b) a set of antecedents consisting of a mixture of assumptions and non-assumption literals.

Using this notion of assumption, it is possible to show the following:

Lemma 1 *Under conditions 1 and 2 above, all minimal support sets consist of assumptions only.*

This restricted support set is called a *label.* More formally, a label for x, $\mathcal{L}(x, \Sigma)$, is given by

$$\mathcal{L}(x, \Sigma) = \{ \bigwedge_{A_j \in \mathcal{A}} A_j \mid (\bigvee_{A_j \in \mathcal{A}} \overline{A_j}) \text{ is a minimal support clause for } x \text{ with respect to } \Sigma\}. \quad (7)$$

Note that a label is a restriction of a minimal support clause to a minimal support clause consisting only of marked literals. If the clauses are propositional Horn, we can represent the label in terms of prime implicates [Reiter and de Kleer, 1987]:

$$\mathcal{L}(x, \Sigma) = \{ \bigwedge_{A_j \in \mathcal{A}} A_j \mid (\bigvee_{A_j \in \mathcal{A}} \overline{A_j} \vee x) \text{ is a prime implicate of } \Sigma\}.$$

The *minimal label set* of a clause is a disjunction of the clause's labels.[7]

A minimal label set can be viewed as a DNF version of a minimal support set. For Horn clauses, the marked literals always appear negated in clauses (i.e. appear un-negated on the LHS of an implication). Hence they are negated in support clauses (CNF), and will appear un-negated in labels.

We conclude this section with a set of definitions which will be used in the description of network reliability algorithms. We call the conjunction of the Σ_i's a Boolean expression[8] F, i.e. $F = \wedge_{i=1,\ldots,l} \Sigma_i$. We note that there may be many other expressions F' which also compute such a Boolean expression F, where an expression F' computes F if $F'(x) = F(x)$ for all instantiations of $\mathbf{x}$. For example, an expression composed of the set of prime implicates derived from the set Σ of clauses in F, F_Π, also computes F. We define the cost of F as the number of clauses in F. An irredundant expression[9] $\mathcal{F}$ is an expression such that $\mathcal{F}$ computes F and no expression computing F has cost smaller than $\mathcal{F}$.

Example 1:

We represent each clause as a disjunction of the literals enclosed in square brackets.

$$\begin{aligned} \Sigma = \{&[\overline{x}_1, \overline{x}_2, x_{12}], [\overline{x}_3, x_4, \overline{x}_5, x_{13}], [\overline{x}_5, x_{14}], [\overline{x}_6, \overline{x}_7, x_8, x_{14}], \\ &[x_7, \overline{x}_8, \overline{x}_{11}, x_{15}], [\overline{x}_9, \overline{x}_{10}, x_{15}], [\overline{x}_{12}, x_{16}], [\overline{x}_{13}, x_{16}], [\overline{x}_{14}, x_{17}], [\overline{x}_{15}, x_{17}], \\ &[\overline{x}_{16}, x_{17}, \overline{x}_{19}], [\overline{x}_{17}, x_{18}], [\overline{x}_2, x_{18}], [\overline{x}_{18}, x_{19}], [\overline{x}_1, x_{17}]\} \end{aligned} \quad (8)$$

[7] Assumptions are assigned to every literal so that Belief measures can be computed for every clause. This can easily be relaxed, but Lemma 1 will no longer hold, i.e. some minimal support sets will consist of assumptions and literals. For such support sets, measures may not be computed.

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

[9] Sometimes referred to as a minimal expression.

$$\Pi(X) = \{[\overline{x}_1, x_{18}], [\overline{x}_1, x_{19}], [\overline{x}_1, x_{17}], [\overline{x}_2, x_{18}], [\overline{x}_2, x_{17}, \overline{x}_{13}], [\overline{x}_2, x_{17}, \overline{x}_{12}], [\overline{x}_2, \overline{x}_{16}, x_{17}], [x_{18}, \overline{x}_5], [x_{17}, \overline{x}_5], [x_{18}, \overline{x}_9, \overline{x}_{10}], [x_{17}, \overline{x}_9, \overline{x}_{10}], [x_{18}, x_7, \overline{x}_8, \overline{x}_{11}], [x_{17}, x_7, \overline{x}_8, \overline{x}_{11}], [x_{18}, \overline{x}_{15}], [\overline{x}_{15}, x_{17}], [x_{18}, \overline{x}_6, \overline{x}_7, x_8], [x_{17}, \overline{x}_6, \overline{x}_7, x_8], [x_{18}, \overline{x}_{14}], [\overline{x}_{14}, x_{17}], [\overline{x}_{17}, x_{18}], [\overline{x}_{18}, x_{17}, \overline{x}_{13}], [\overline{x}_{18}, x_{17}, \overline{x}_{12}], [\overline{x}_{18}, \overline{x}_{16}, x_{17}], [x_{19}, \overline{x}_2], [x_{19}, \overline{x}_5], [x_{19}, \overline{x}_9, \overline{x}_{10}], [x_{19}, x_7, \overline{x}_8, \overline{x}_{11}], [x_{19}, \overline{x}_{15}], [x_{19}, \overline{x}_6, \overline{x}_7, x_8], [x_{19}, \overline{x}_{14}], [x_{19}, \overline{x}_{17}], [\overline{x}_{18}, x_{19}], [x_{18}, \overline{x}_{19}, \overline{x}_{13}], [x_{18}, \overline{x}_{19}, \overline{x}_{12}], [x_{18}, \overline{x}_{16}, \overline{x}_{19}], [x_{17}, \overline{x}_{19}, \overline{x}_{13}], [x_{17}, \overline{x}_{19}, \overline{x}_{12}], [\overline{x}_{16}, x_{17}, \overline{x}_{19}], [x_{16}, \overline{x}_3, x_4, \overline{x}_5], [\overline{x}_{13}, x_{16}], [x_{16}, \overline{x}_1, \overline{x}_2], [\overline{x}_{12}, x_{16}], [\overline{x}_9, \overline{x}_{10}, x_{15}], [x_7, \overline{x}_8, \overline{x}_{11}, x_{15}], [\overline{x}_6, \overline{x}_7, x_8, x_{14}], [\overline{x}_5, x_{14}], [\overline{x}_1, \overline{x}_2, x_{12}], [\overline{x}_3, x_4, \overline{x}_5, x_{13}]\}$$

The minimal support set for given clauses is as follows:

Σ_i	$\xi(\Sigma_i, \Sigma)$
$[\overline{x}_1]$	$[x_{17}], [x_{18}], [x_{19}], [\overline{x}_2, x_{16}], [\overline{x}_2, x_{12}]$
$[\overline{x}_9, \overline{x}_{10}]$	$[x_{17}], [x_{18}], [x_{19}], [x_{15}]$
$[\overline{x}_8, \overline{x}_{11}]$	$[x_7, x_{17}], [x_7, x_{18}], [x_7, x_{19}]$
$[\overline{x}_6, \overline{x}_7]$	$[x_8, x_{17}], [x_8, x_{18}], [x_8, x_{19}], [x_8, x_{14}]$
$[\overline{x}_7, x_8]$	$[\overline{x}_6, x_{14}], [\overline{x}_6, x_{17}], [\overline{x}_6, x_{18}], [\overline{x}_6, x_{19}]$
$[\overline{x}_3, x_4]$	$[\overline{x}_5, x_{13}], [\overline{x}_5, x_{16}]$

Example 2:
Consider the following example in which assumptions are assigned to each clause.

$$\Sigma = \{[\overline{A}_1, x_1], [\overline{A}_2, x_4], [\overline{x_1}, \overline{A_3}, x_2], [\overline{x_2}, \overline{A_4}, x_3], [\overline{x_1}, \overline{A_5}, x_4], [\overline{x_4}, \overline{A_6}, x_5], [\overline{x_2}, \overline{x_4}, \overline{A_7}, x_5]\}$$

The minimal label sets assigned to the literals are:

LITERAL	LABEL SET
x_1	$\{A_1\}$
x_2	$\{A_1, A_3\}$
x_3	$\{A_1, A_3, A_4\}$
x_4	$\{\{A_2\}, \{A_1, A_5\}\}$
x_5	$\{\{A_2, A_6\}, \{A_1, A_5, A_6\}, \{A_1, A_2, A_3, A_7\}, \{A_1, A_3, A_5, A_7\}\}$

4 A LOGIC-BASED FORMULATION OF DEMPSTER SHAFER THEORY

In this section we show the logical analogs of various DS theory functions. We first define the correspondence of set-theoretic and logic-theoretic notions in order to understand the relationship between the traditional set-theoretic description of DS theory and the logi-theoretic

description; we also discuss the role of the normalization function in DS theory. We then discuss the two distinct Boolean operations which are necessary to compute the DS Belief function for a proposition x within a logical setting: (1) computing the label set for x $\mathcal{L}(x, \Sigma)$; and (2) computing a disjoint Boolean expression from $\mathcal{L}(x, \Sigma)$.

4.1 Logic and Set Theoretic Definitions

Given the definitions of what DS theory computes in Section 2, we now show their logical interpretation. We first need to describe the correspondence between set theory and logic. Such a correspondence has been known for a long time, and in particular, the logical relationships of the set theoretic operations underlying different uncertainty formalisms has been carefully studied. For example, Carnap [1962] and de Finetti ([1974], p. 37 ff.) have analysed the logical foundations of probability theory. Similarly, Shafer [1976] implicitly defined a correspondence between set-theoretic notions relevant to subsets of Θ and logical notions. More precisely, as described by Shafer ([1976], p. 37), we formulate the following definition.

Definition 1 *If θ_1 and θ_2 are two subsets of Θ and Σ_1 and Σ_2 are the logical propositions corresponding to θ_1 and θ_2 respectively, then the set theoretic notions hold if and only if the corresponding logic theoretic notions hold, as shown in Table 2.*

Table 2: Correspondence of set theoretic and logic theoretic notions

SET THEORETIC	LOGIC THEORETIC
$\theta_1 \cap \theta_2$	$\Sigma_1 \wedge \Sigma_2$
$\theta_1 \cup \theta_2$	$\Sigma_1 \vee \Sigma_2$
$\theta_1 \subset \theta_2$	$\Sigma_1 \Rightarrow \Sigma_2$
$\theta_1 = \overline{\theta_2}$	$\Sigma_1 = \neg\Sigma_2$

In Table 2 $\theta_1 = \overline{\theta_2}$ means that θ_1 is the set-theoretic complement of θ_2.

4.2 The Question of Normalization

The denominator of the RHS of Dempster's Rule of Combination (cf. Equations 4 and 6) is a normalization function. We ignore this normalization function in the following logical analysis of Dempster Shafer Theory for two reasons. First, the normalization function has been shown to be irrelevant [Hummel and Landy, 1988]. Hummel and Landy describe a state of belief in DS theory, as defined by a bpa ϱ, in terms of $(\mathcal{M}, \oplus)$, where $\mathcal{M}$ is a monoid and $\oplus$ is the (normalized) combination operation. They show that an unnormalized space of belief states $(\mathcal{M}', \oplus')$ can be homomorphically mapped onto $(\mathcal{M}, \oplus)$, and is less cumbersome and more easily understood. All operations can be done in $(\mathcal{M}', \oplus')$-space without loss of generality,

and can be mapped into the original $(\mathcal{M}, \oplus)$-space if necessary. For the purposes of this paper, ignoring the normalization function simplifies the discussion and does not involve any loss of generality, as the normalization function can be modeled in logical terms by a simple extension of our analysis.

Second, the normalization function is controversial because there are some situations in which it gives counter-intuitive results. One case is the Three Prisoner's paradox, as discussed at length by Pearl [1988]. A second case arises when pooling near-contradictory evidence. Zadeh [1984] demonstrates the counter-intuitive results which can be obtained in such cases. We briefly discuss the reasons for these counter-intuitive results in Section 8.

Using the normalization function can be thought of as adopting a closed word assumption,[10] in which the frame of discernment Θ is exhaustive, including all possible propositions, and excluding no propositions relevant to Θ. Shafer [1976], among others, adopts the closed world assumption. The converse of a closed world assumption, an open world assumption, ignores the normalization function by assuming that Θ is not exhaustive. Smets [1988] and Hummel and Landy [1988] advocate an open world assumption.

The choice of a closed or open world assumption affects the measures assigned following Belief updates. A closed world assumption ensures that, following evidential updates, the total mass assigned to the consistent propositions has measure 1. This is guaranteed by the use of a normalisation function. An open world assumption, in contrast, entails assignment of a measure of less than 1 to the consistent propositions, because of the assignment of increasing mass to the empty set $\emptyset$ as more contradictions are discovered. Consequently, the Belief (and Plausibility, Commonality, etc.) measures decrease. Implementations based on an open world assumption are subject to roundoff error as Belief assignments approach zero. However, they avoid the counter-intuitive results introduced by the normalization function when pooling near-contradictory evidence.[11]

4.3 Logical Correspondence of Dempster Shafer Theory

In this section we make the logical correspondence of DS theory explicit, and use it to compare and contrast the manipulation of DS Belief functions with certain logic-theoretic manipulations. We note that certain aspects of DS theory which do not occur in logic require extensions to traditional logic. These include

- Two arbitrary propositions (e.g. θ_i and θ_j) in DS theory can be defined (external to the logic) as being contradictory.[12] This is equivalent to two arbitrary logical clauses (e.g. Σ_i and Σ_j) being contradictory.

- DS theory can be used to pool multiple bodies of evidence. Since Dempster's rule is commutative, this pooling can be done dynamically, and in any order. Logical resolu-

[10] The closed world assumption has been formalized logically by Reiter [1980].

[11] However, they do not avoid the problems encountered with paradoxes like the Prisoners' dilemma. The implies that there are cases in which DS theory is inappropriate.

[12] More precisely, they can be defined to be mutually exclusive.

tion is typically considered not to be a dynamic process, in that the set of clauses to be resolved typically does not change dynamically. In other words, logic traditionally assumes a fixed set of clauses. We show the changes necessary to update a database consisting of propositional logic clauses.[13]

We now show the correspondence of set theoretic notions and propositional clauses, of symbolic Belief functions and minimal support clauses, and of the Belief function combination rule $\oplus$ and minimal support clause combination.

We start out by defining a set of DS Theory focal propositions $\Theta = \{\theta_1, ..., \theta_n\}$ and corresponding logical clauses $\Sigma = \{\Sigma_1, ..., \Sigma_n\}$. Thus, within the Boolean algebra $\mathcal{B}$, a subset $\mathcal{D} \subset \mathcal{B}$ of mutually exclusive and exhaustive propositions are defined as the frame of discernment. To each focal proposition there is a measure assigned, which produces a set of n bpa's $\{\varrho_1, \varrho_2, ..., \varrho_n\}$. In the logical framework, assumption A_i is the symbolic representation for ϱ_i, and $\mathcal{A} = \{A_1, ..., A_n\}$ is the symbolic representation of the n bpa's $\{\varrho_1, \varrho_2, ..., \varrho_n\}$. The logic-based version of measure assignment to $\varrho : 2^{\Theta} \rightarrow [0,1]$ is $\varrho : 2^{\mathcal{A}} \rightarrow [0,1]$. Thus, we might have the clauses and associated bpa's:

$$\begin{array}{lll} \overline{x_1} \vee \overline{x_2} \vee \overline{A_1} \vee x_5 & \varrho_1(A_1) = 0.6 & \varrho_1(\overline{A_1}) = 0.4 \\ \overline{x_2} \vee \overline{x_4} \vee \overline{A_2} \vee x_6 & \varrho_2(A_2) = 0.8 & \varrho_2(\overline{A_2}) = 0.2. \end{array}$$

Given this framework, we first define how to evaluate the mass assigned to a support clause:

Definition 2 *The measure assigned to a minimal support clause, $\xi(\Sigma_k, \Sigma)$, is given by*

$$\varrho(\xi(\Sigma_k, \Sigma)) = \prod_{A_j \in \xi(\Sigma_k, \Sigma)} \varrho(A_j). \tag{9}$$

For example, for a support clause $\xi(x_7, \Sigma) = \overline{A_2} \vee \overline{A_4}$, we have $\varrho(\xi(x_7, \Sigma)) = \varrho(A_2) \cdot \varrho(A_4)$.

We now show that the support clause for a literal is equivalent to a symbolic representation of the mass assigned to that literal.

Lemma 2 *The belief assigned to a proposition θ (which has corresponding logical clause Σ_k) can be computed from the minimal support clause for Σ_k; i.e.*

$$Bel(\theta) = \varrho(\xi(\Sigma_k, \Sigma)).$$

We note that all logic-theoretic correspondences to $\varrho(\theta)$ and $Bel(\theta)$ are Boolean expressions which, in general, are not necessarily disjoint. A DNF Boolean formula is disjoint if each pair of conjunctive clauses is disjoint. A pair of conjunctive clauses are disjoint if, for each variable common to the clauses, say x_j, one clause contains the variable and the other contains the negated variable $\overline{x_j}$.

[13] We will use the fact that since F is also computed by F_{Π}, one only needs to maintain F_{Π}, and can update Π and "ignore" F.

Definition 3 *Given a CNF Boolean expression F consisting of a set of disjuncts $F_1, ..., F_q$, a* **disjoint CNF Boolean expression** *$F' = disj(F))$ computed from an expression F consists of a set of disjoint disjuncts $F'_1, ..., F'_l$ such that $disj(F)$ computes F.*

An analogous definition exists for DNF expressions.

A disjoint expression for proposition x is necessary to evaluate the correct mass or Belief assigned to x. If it is not disjoint, the Boolean expression must be expanded until it is. Shafer [1982] noted this assumption of conditional independence in the combination of "coded messages" (from which Belief functions are constructed).

The label sets assigned to clauses are in DNF form (cf. equation 7). The measure assigned to the expression for a label set can be evaluated as follows.

Definition 4 *The measure assigned to a DNF formula*

$$F = \bigvee_{F_j \in F} \bigwedge_{A_i \in F_j} A_i$$

is given by

$$\varrho(F) = \bigvee_{F_k' \in disj(F)} \bigwedge_{A_i \in F_k'} \varrho(A_i).$$

A well-known example of the need for disjointness to add uncertainty measures for unions of events is the probabilistic restriction (Bayesian Belief functions). In this case, it is well known that

$$Prob\{A \cup B\} = Prob\{A\} + Prob\{B\} - Prob\{A\}Prob\{B\}, \tag{10}$$

and that

$$Prob\{A \cup B\} = Prob\{A\} + Prob\{B\} \tag{11}$$

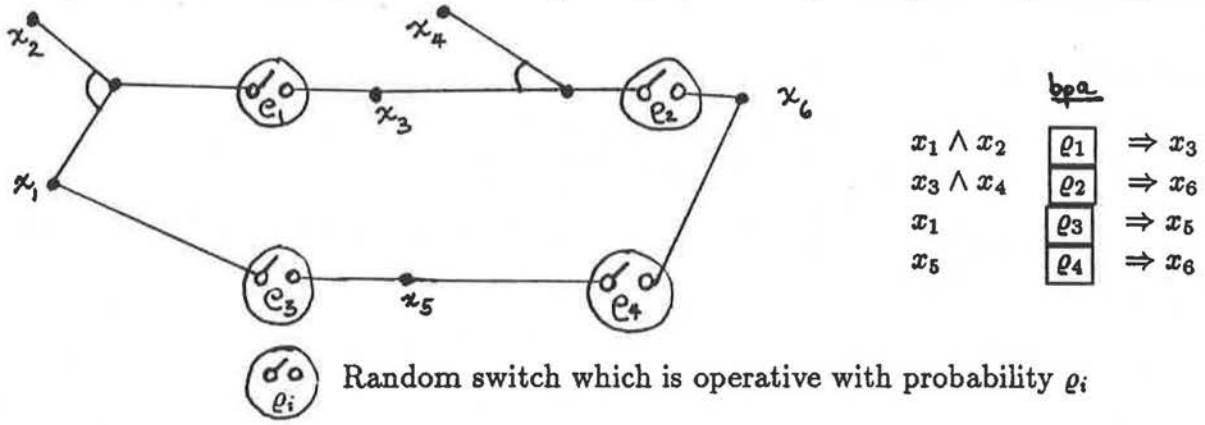
only if $A \cap B = \emptyset$, i.e. A and B are disjoint.

The expansion of a Boolean expression to its disjoint form corresponds to a Network Reliability computation, and is described in Section 4.4.2. Hence the right-hand-side of equation 10 is the disjoint expansion of the left-hand-side; if A and B are not disjoint, substituting probabilities into the expression on the right-hand-side of equation 11 will give the incorrect answer.

In DS Theory, Belief function combination is done according to Dempter's Rule of Combination (equation 6), and is summarised as $Bel(\theta) = \oplus_i Bel_i(\theta)$. The numerator of Dempster's Rule can be thought of as summing the disjoint proof paths in the proof for θ. Pearl [1988] uses the analogue of a random switch, which assigns mass for a fraction $0 \leq p \leq 1$ of the time and no mass for the fraction $(1 - p)$ of the time. The fraction of time the switch is active gives the probability that a proof path remains uninterrupted. For a literal with many proof paths, the probability assigned to the literal is the sum of the proof path probabilities. However, in order to sum the individual proof path probabilities to find the probability that a literal is provable, pairwise independence of the paths is necessary. Figure 2 shows a random switch model for a proof graph.[14] In the figure $Prob(x_6)$ can be computed by summing its two

[14]This example is taken from [Pearl, 1988].

Figure 2: Random switch model of a proof graph corresponding to the given logical clauses



proof paths, giving the probability $(\varrho_1\varrho_2 + \varrho_3\varrho_4)$. If $\varrho(x_5)$ is defined for the literals $\{x_3, x_5\}$, then the proof paths for x_6 are no longer disjoint, as there is a shared sub-path. Hence, the measure assigned to x_6 is not given by the proof path-set $\varrho_6 = \varrho_1\varrho_2 + \varrho_1\varrho_5\varrho_4 + \varrho_3\varrho_4$, but by the disjoint version of the proof path-set. The derivation of this disjoint form is given in Appendix A.

We now show that in addition to computing mutually independent activity times for proof paths, mass function or Belief function combination can also be explained in terms of support clause computation.

Lemma 3 *Dempster's rule for Belief combination, i.e.*

$$Bel(\theta) = \frac{\sum_{\bigcap_i \theta_i = \theta} Bel_1(\theta_1)Bel_2(\theta_2)\cdots Bel_m(\theta_m)}{1 - \sum_{\bigcap_i \theta_i = \emptyset} Bel_1(\theta_1)Bel_2(\theta_2)\cdots Bel_m(\theta_m)}, \tag{12}$$

corresponds to computing $Bel(\Sigma_j)$, the measure assigned to a disjoint form of $\xi(\Sigma_j, \Sigma)$, where Σ_j is the clause corresponding to θ.

Note that in the more general case in which arbitrary θ_i and θ_j may provide conflicting information, DS Theory cannot assign weight to a "contradiction", and so all information pooling must be done over non-conflicting subsets. In a strict logic-based formulation, the fact that x_i and x_j are contradictory must be explicitly encoded as a clause so that their conjunction cannot be created in any support clause. Hence the following clause can be created: $\neg(x_k \wedge x_l)$ which is equivalent to $(\overline{x_k} \vee \overline{x_l})$. In the formulation involving assumptions, contradictory propositions θ_i and θ_j can be modeled by the corresponding assumptions A_i and A_j being contradictory, i.e. $A_i \wedge A_j \Rightarrow \top$, where $\top$ denotes a contradiction. For conflicting information,

Dempster's Rule of Conditioning can be used to condition on the non-contradictory evidence. Dempster's Rule of Conditioning is as follows:

Lemma 4 *Suppose Bel' is given by*

$$Bel'(\theta_1) = \begin{cases} 1 & if\ \theta_2 \subset \theta_1 \\ 0 & if\ \theta_2 \not\subset \theta_1 \end{cases}$$

and Bel_1 is another belief function over Θ. If Bel and Bel' are two combinable Belief functions, let $Bel(\cdot|\theta_2)$ denote $Bel \oplus Bel'$. Then

$$Bel(\theta_1|\theta_2) = \frac{Bel(\theta_1 \cup \overline{\theta_2}) - Bel(\overline{\theta_2})}{1 - Bel(\overline{\theta_2})} \tag{13}$$

for all $\theta_1 \subset \Theta$.

Hence if θ_2 is a contradictory proposition, then conditioning on $\overline{\theta_2}$ is done. No normalization is necessary for an open world assumption. Note that equation 13 is a special case of equation 6 with $Bel'(\theta_2)$=1, i.e. under Bel' θ_2 is known to be true.

The assumption of a closed world requires a simple extension of the preceding discussion, a renormalisation of the Belief functions based on the weight assigned to the null set $\emptyset$ (i.e. to contradictions). In this closed world case it is assumed that no weight is assigned to unknown propositions. Hence, whenever contradictory propositions are assigned mass, this mass must be assigned to $\emptyset$, and all Belief assignments renormalised to ensure that the total mass has measure 1.

Given these Boolean expressions, they must now be made disjoint. We discuss this process in the next section. Once a disjoint Boolean expression has been obtained, the Belief measure can be obtained by substituting in the measures for the basic probability assignments.

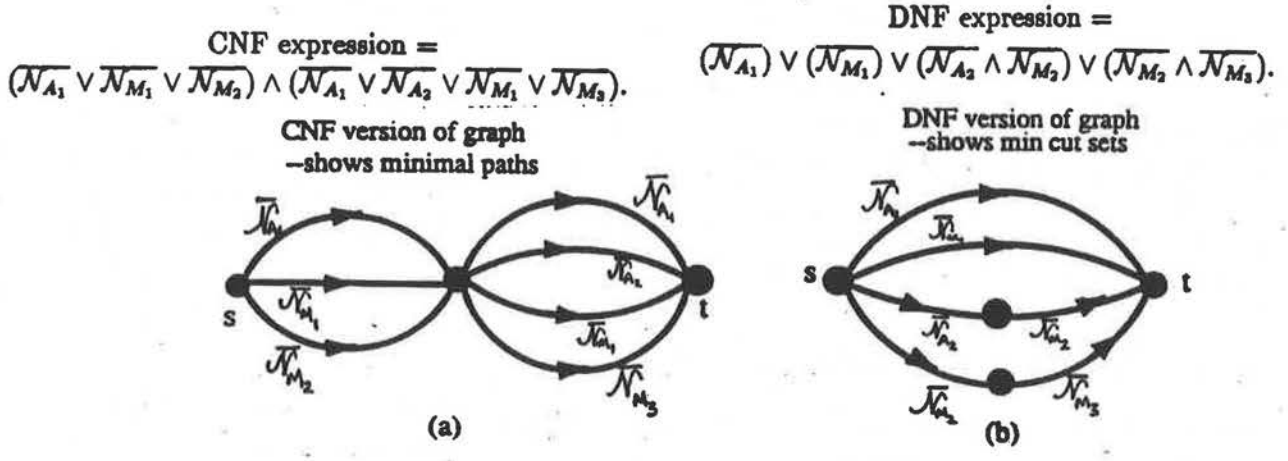
4.4 Computing Disjoint Boolean Expressions

The process of making Boolean expressions disjoint can be better understood in graph theoretic terms. Note that this graph theoretic notation is a restriction of the hypergraph notation (cf. §2) to hyperedges with two vertices. We describe the necessary notation in the following section. We also use this notation to describe the isomorphism between ensuring the disjointness of expressions for DS Belief functions and for network reliability measures, and more generally the isomorphism between Belief function updating and the computation of network reliability measures.

4.4.1 Graph Theoretic Notation

To exploit the DS theory/graph theory isomorphism, we use the following well-known correspondence:

Figure 3: Graphs corresponding to circuit diagnosis formulae



Lemma 5 *Any Boolean expression F has an associated graph $\mathcal{G}(V, E)$ consisting of vertices V and edges E.*

There are several methods of constructing $\mathcal{G}$ from F. For example, we may assume a Boolean literal x_i to correspond to an edge E_i, and a logical connective $(\vee, \wedge)$ to correspond to a vertex that joins two or more edges between the corresponding components as follows: an $\wedge$ connecting two literals (or clauses) corresponds to an edge connecting two vertices (or vertex sets) in series, and an $\vee$ 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 $\mathcal{P}$ 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.

There are two additional well-known correspondences between a graph $\mathcal{G}$ and the corresponding set of clauses Σ. The first is for an expression F expresses in CNF.

Lemma 6 *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 7 *The set of prime implicates $\Pi(\Sigma)$ for a DNF Boolean expression F defines the cut sets of the corresponding graph $\mathcal{G}$.*

Example:

The graph corresponding to the irredundant CNF expression for the circuit diagnosis example described in Section 6.2 is shown in Figure 3(a). The CNF expression is

$$(\overline{N_{A_1}} \vee \overline{N_{M_1}} \vee \overline{N_{M_2}}) \wedge (\overline{N_{A_1}} \vee \overline{N_{A_2}} \vee \overline{N_{M_1}} \vee \overline{N_{M_3}}). \quad (14)$$

The DNF version of equation 14 is obtained by standard CNF to DNF conversion techniques. We emphasize that these graphs are interconvertible. By finding the edge cuts containing the minimum number of edges for the graph shown in Figure 3(a), we can obtain a minimal cut representation shown in Figure 3(b). Note the graph theoretic relationships between the diagnostic notions[15] of minimal conflict sets (minimal cut sets) and minimal candidates (minimal paths).

4.4.2 Network Reliability Computation

Network Reliability describes a set of techniques for analysing computer and communication networks. The network reliability problem computes the probability that the network (or a portion of the network) is functioning. The input to a network reliability algorithm is (1) a Boolean expression F (which describes a network in which each literal represents a network component) and (2) a [0,1] assignment of weights to Boolean variables, which corresponds to the probability that the component x is functioning.

Network reliability is a restriction of DS theory to Bayesian Belief functions. In logical terms, each clause consists of two literals, and hence a network reliability problem is an instance of 2SAT, the SATISFIABILITY problem with two literals per clause.

If we frame this problem in graph theoretic terms, the weighted Boolean expression corresponds to a weighted graph. For a general DS theory problem, we have a weighted hypergraph. Hence, the network reliability problem in graph theoretic terms corresponds to computing the probability that a set of vertices can communicate with one another (i.e. the probability that a path (or set of paths) exists between the specified vertices). The set of support for a proposition x corresponds to the set of paths in the graph to x (for F expressed in DNF), or the cutsets which disconnect x from the graph (for F expressed in CNF). Hence it is obvious that network reliability measures and Bayesian Belief functions compute exactly the same thing: both compute the probability that a (proof) path to a proposition exists in a graph.

A disjoint Boolean expression $disj(F)$ and the equivalent DS Belief function (or system reliability) formula are termwise identical; the operations necessary to convert $disj(F)$ to a DS Belief function formula are given in Table 3.

We note this correspondence between computing DS Belief functions and computing network reliability measures because the latter problem has been carefully studied for many years, and we will use results derived in the network reliability literature in Section 5.

Several methods have been developed for computing network reliability. The computational approaches fall into three categories:

1. Path/cutset enumeration methods;

2. Pivotal factoring/decomposition methods; and

[15] See section 6.2 for a short description of diagnostic reasoning.

Table 3: Correspondence of Boolean and Network Reliability Forms for disjoint Boolean formulae

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

3. Topological decomposition methods.

Each approach simply ensures pairwise disjointness of the network reliability expression. For example, the path/cutset enumeration approach ensures that, in an expression consisting of all pathsets/cutsets in a graph, no pair of paths/cutsets has an overlap. An overlap of two paths means that there is a shared sub-path, rendering the two paths non-disjoint. We note that network reliability can be computed directly from the graph $\mathcal{G}$ or from the paths or cutsets of $\mathcal{G}$. We discuss each of these techniques in Appendix A.

4.5 Belief Function Algorithm

We now describe a Belief function algorithm in terms of the logical operations we have defined. The input is the tuple $(\Sigma, \mathcal{A}, \varrho)$, and a clause Σ_k for which we want to compute a Belief function. In implementing Dempster's rule within a logical perspective, it is important (1) to ensure null intersection with contradictory propositions, as Dempster's rule sums over only consistent propositions, and (2) to ensure disjoint Boolean expressions. The algorithm for implementing Dempster's rule is as follows:

1. Compute the label for Σ_k, $\mathcal{L}(\Sigma_k, \Sigma)$. This is done by (1) computing the set $\Pi(\Sigma)$ of prime implicates, (2) from $\Pi(\Sigma)$ determining the set of support for Σ_k, and, if necessary, (3) converting the set of support to a label set. We refer to the label set as a disjunction of labels $\mathcal{L}_i(\Sigma_k, \Sigma)$, i.e. $\mathcal{L}(\Sigma_k, \Sigma) = \bigvee_i \mathcal{L}_i(\Sigma_k, \Sigma)$, where each $\mathcal{L}_i = \bigwedge_j A_j$.

2. Account for contradictions, which we call Φ. A contradiction consists of a conjunction of two or more sets of propositions, where Φ is the set of contradictions. The contradiction-free Boolean expression required is $\mathcal{L}(\Sigma_k, \Sigma) \cap (\neg\Phi)$

3. Compute a disjoint Boolean expression, i.e. $disj(\mathcal{L}(\Sigma_k, \Sigma) \cap (\neg\Phi))$.

4. Compute the Belief assignment, using Definition 4:

$$\begin{aligned} Bel(\Sigma_k) &= \varrho(disj(\mathcal{L}(\Sigma_k, \Sigma) \cap (\neg\Phi))) \\ &= \bigvee_{Z_l \in disj(\mathcal{L}(\Sigma_k,\Sigma)\cap(\neg\Phi))} \bigwedge_{A_j \in Z_l} \varrho(A_j) \end{aligned}$$

5. Substitute mass functions for the A_i's to calculate the Belief function for Σ_k using Table 3.

If normalization is required, then the normalization, as given by equation 6, where the normalized Belief expression is $K^{-1}Bel(\Sigma_k)$, and K is given by

$$\begin{aligned} K &= 1 - \sum_{\bigcap_i \theta_i = \emptyset} Bel_1(\theta_1)Bel_2(\theta_2)\cdots Bel_m(\theta_m) \\ &= 1 - Bel\{\Phi\} \\ &= 1 - \bigvee_{Z_l \in disj(\Phi)} \bigwedge_{A_j \in Z_l} \varrho(A_j) \end{aligned}$$

This operation can be considered to be computing Belief by conditioning on the absence of contradictions, using Dempster's Rule of Conditioning:

$$Bel(\Sigma_k|\neg\Phi) = \frac{Bel(\Sigma_k \cup \Phi) - Bel(\Phi)}{1 - Bel(\Phi)}$$

We note that normalisation is not necessary for an open world assumption.

An example of the operation of this algorithm, as implemented within an ATMS, is presented in Section 6.2.

5 COMPUTATIONAL COMPLEXITY

5.1 Definition of Problems

We now define the complexity of (1) the overall problem of computing Belief functions, and (2) that of the subproblems of the DS Belief function algorithm presented in the previous section. Because computational complexity is not the main focus of this paper, we will cite results obtained elsewhere. Instead we focus on discussing the implications of these results. A full treatment of these complexity issues can be found in [Provan, 1990b] and [Provan, 1988b].

The complexity of updating using Dempster's rule, e.g. computing exact DS Belief functions for multiple bpa's, has not been closely studied beyond noting that such a computation is exponential in the size of the frame of discernment Θ.[16] The number of subsets of Θ increases exponentially with $\mid \Theta \mid$, and the normalizing function can sum over all of these subsets, so computing a single normalization function can be computationally expensive.

In the following discussion of complexity results, we assume familiarity with the concepts of P, NP and NP-completeness.

Given a single bpa, computing the Belief (Plausibility, etc.) assigned to a proposition, the problem $DSb(\Theta, \varrho, \theta)$, is linear in the size of the frame of discernment, i.e. $O(|\Theta|)$.[17]

[16] Barnett [Barnett, 1981] stated that the complexity of the problem grows exponentially in the number of evidential sources, but never proved this complexity result, a point he noted in a footnote in [Barnett, 1981].

[17] Actually, it is linear in the number of measures assigned by the bpa.

This is because disjointness is assured by every pair of measures in the bpa being mutually exclusive and independent. For multiple bpa's, this disjointness is not certain. The number of subsets which must be counted to pool evidence is also greater than in the case of a single bpa. Using these intuitions, Theorem 1 shows the combination of evidence using Dempster's rule (equation 6) to be the computationally expensive aspect of DS theory.[18]

Theorem 1 *It is a #P-complete problem to compute the DS WEIGHT ASSIGNMENT* $[DSM(\Theta, \vec{\varrho}, \theta)]$ *and DS BELIEF ASSIGNMENT* $[DSB(\Theta, \vec{\varrho}, \theta)]$ *functions.*

Because Dempster's rule is a #P-complete function, it is unlikely that a polynomial-time algorithms exists for this function.[19] Hence, it is unlikely that the logic-based algorithm just presented is polynomial-time. However, it might be that, given the labels for the propositions, the Belief computations are simple. This turns out not to be the case. We formalize this as follows.

The Belief function algorithm can be divided into two main steps: (1) computing the label set, and (2) computing the Belief assigned to a proposition, which necessitates computing the disjoint Boolean expression from the label for the proposition. The corresponding sub-problems are defined as follows:

$\underline{DSB_{\mathcal{L}}}$ (Dempster Shafer Belief Label Computation)

Given a set Σ of clauses, determine the set of labels $\mathcal{L}$ for the database literals.

$\underline{DSB_{D}}$ (Dempster Shafer Belief *D*isjointness computation)

Given $(\mathcal{L}, \vec{\varrho})$, compute the disjoint Belief assigned to a literal x or clause Σ_k.

We now define the complexity of these two problems. Lemma 8 provides upper and lower bounds for almost all Boolean expressions[20] [Provan, 1988b].

Lemma 8 $(DSB_{\mathcal{L}})$ *Generating the label set for a set* $\mathbf{x}$ *of literals with respect to a set* Σ *of clauses is of complexity exponential in the number* n *of literals for almost all propositional expressions* F.

[18] The proof for this theorem is given in [Provan, 1990b]. Orponen [Orponen, 1989] has independently proven this result.

[19] Intuitively, the class #P contains a set of enumeration problems. For example, the enumeration problem associated with SATISFIABILITY is to compute the number of satisfying assignments. A #P-complete function f is one which belongs to the class #P [Valiant, 1979], and every other function in #P can be computed by a deterministic polynomial time Turing machine using f as an oracle. The class #P is at least as intractable as the class NP, and contains several enumeration problems the decision versions of which are NP-complete, such as SATISFIABILITY, CLIQUE, and HAMILTONIAN CIRCUIT [Valiant, 1979]. Completeness of a problem $\mathcal{P}$ for the complexity class #P indicates that $\mathcal{P}$ is more intractable than an NP-complete problem, since #P contains harder problems than NP.

[20] A 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 $n \rightarrow \infty$. See [Zhuravlev, 1982] for details.

This lemma states that almost all expressions have the same order of growth as for the most complex expression, i.e. that almost all Boolean expressions must have a label set whose size is exponential in the number of literals. Hence, for almost all problems, determining a $\Theta(2^n)$ label set will take $\Theta(2^n)$ time, and storing such alabel set will take $\Theta(2^n)$ space.

DSB_D are of worst-case complexity exponential in the number of vertices (i.e. of underlying literals in the corresponding Boolean expression) or paths (i.e. of prime implicates π,[21] in the corresponding Boolean expression) [Provan and Ball, 1984]. Furthermore, in practice the largest networks which can be solved in a reasonable amount of time contain on the order of 50 vertices ([Ball, 1986],[Provan, 1986]). This can be stated as follows:

Lemma 9 (DSB_D) *Generating a disjoint expression from the label set $\mathcal{L}(\Sigma)$ computed from the original expression $F = \bigwedge_i \Sigma_i$ is of complexity exponential in the number n of literals or m of labels.*

In terms of computing DS Belief functions, this means that even given the label set of support associated with a set of database literals, computing the DS Belief for these literals is unlikely to be of complexity polynomial in the number of literals unless $P = NP$.

5.2 Discussion

We have shown in Section 5 that the problem of computing DS Belief functions (assuming that Belief combination is required) to be intractable. In addition, the logic-based method proposed in this paper consists of two intractable steps, label generation and disjoint expression computation. For cases in which the support sets are used for purposes other than the calculation of DS Belief functions, e.g. using the support sets to facilitate diagnostic reasoning [de Kleer and Williams, 1987] this is a reasonable approach. However, if Belief function computation from a database of logical clauses is the primary objective, then the use of some Network Reliability algorithm is more efficient, because computing the label set $\mathcal{L}$ for a database[22] and computing Belief functions from $\mathcal{L}$ is less efficient than computing the Belief functions directly. The list of negative complexity results (from the point of view of the existence of polynomial-time algorithms) concurs with reports of practical experience ([Ball, 1986], [Provan, 1986]).

Hence it appears that it is unlikely that there exist polynomial-time algorithms, or that large network reliability problems can be exactly solved efficiently. This implies that computing Belief functions for even moderately-sized frames of discernment cannot be done efficiently. This necessitates the examination of tractable restrictions of Dempster Shafer theory.

5.3 Tractable Transformations

There have been two approaches to transforming DS Theory to improve computational tractability. The first restricts the domain over which computation takes place, and the second is based

[21] A prime implicate π for a propositional Horn expression can be defined as the disjunction of a literal x and the label for x, i.e. $\pi = x \vee \mathcal{L}(x)$.

[22] This computation, as shown by Lemma 8, is exponential in the size of the database.

on hypertree embeddings. Barnett [1981] has restricted the domain of computation to focal propositions and their negations, instead of the entire power set of Θ. This ensures a linear algorithm in the number n of focal propositions. The disadvantage is the loss of expressiveness.

The second type of transformation, the embedding of an instance of a DS Theory problem within a hypertree, has been studied by many researchers, including Shenoy and Shafer [1988] and Shenoy, Shafer and and Mellouli [1986]. This hypertree restriction enables propagation of Belief functions based on local computations, similar to the local computations for propagation within Bayesian networks [Pearl, 1986] and Influence Diagrams ([Howard and Matheson, 1981],[Shachter, 1986]).

This hypertree model is important because computations within a hypertree are efficient, as opposed to similar computations being intractable within a general hypergraph. Thus, if a general DS theory problem can be embedded in a hypergraph, the problem becomes easy. However, even though finding any hypertree embedding is easy, such an operation can increase the size of the hypergraph by an exponential factor. Finding a good hypertree embedding is NP-hard, and has been studied by several researchers, including [Arnborg *et al.*, 1987], [Tarjan and Yannakakis, 1984], and [Zhang, 1988]. Both types of transformations are studied in more detail in [Provan, 1990b].

5.4 Approximation Methods for Full Dempster Shafer Theory

Several approximation methods have been studied within the network reliability literature. Their goal is to avoid the intractability associated with computing exact reliability formulae (or exact DS Belief functions). In cases for which bounded approximations are sufficient, polynomial-time or linear-time algorithms can be used for Belief function computation. A DS Belief function is itself an uncertainty measure, and for many applications an increase in the "level" of uncertainty may not affect the outcome. For example, if all that is required is a rank-ordering of sets of propositions, an approximation which preserves relative rank will be sufficient.

Describing the many approximation methods is beyond the scope of this paper. We outline a few methods, and refer the reader to [Provan, 1990b] for more detail. We cite results for network reliability, but the methods hold for DS Belief computations as well.

A variety of approximation methods (for which no theoretical analysis exists) have been proposed within the network reliability literature. These methods can be described, on the whole, as "quick and dirty", as they are based on heuristic rather than theoretical arguments. We present four examples of such methods.

Provan [1986] outlines criteria for good approximation methods for reliability computations, and also demonstrates how these criteria can be applied to specific classes of network reliability problems. However, it has been shown [Provan and Ball, 1983] that it is unlikely that polynomial-time deterministic approximation algorithms exist. It is possible that randomized approximation methods can be developed, based on the method proposed by Jerrum and Sinclair [1988].

6 IMPLEMENTATIONS OF DEMPSTER SHAFER THEORY

Implementations exist both of full and of restricted cases of DS Theory. However, because of the computational complexity associated with computing Belief functions, many of these implementations are restrictions of DS theory. We focus primarily on a description of the implementation of DS Theory based on logic, as the relation between logic and DS Theory is the primary objective of this paper. To date, the logic-oriented implementations of *full* DS theory are all based on the Assumption-based TMS (ATMS) of de Kleer [1986].

6.1 Logic-Based Implementations of Restrictions of Dempster Shafer Theory

Barnett [1981] has implemented an algorithm in PROLOG called Support Logic Programming (SLP) which restricts the domain of computation to Belief functions just for the focal propositions and their negations. This implementation is linear in $|\Theta|$, but significantly restricts the domain of inference.

d'Ambrosio [1987] has also implemented a restricted form of DS theory based on the Evidential Support Logic Programming of Baldwin [1985]. d'Ambrosio attaches Dempster Shafer uncertainty bounds, i.e. $[Bel, Pls]$, to ATMS labels. This approach evaluates the DS Belief (and Plausibility) functions after the ATMS has symbolically determined the set of support for all database literals. Like SLP, this implementation is restricted to assigning Belief only to focal literals and their negations.

6.2 ATMS-based Implementations of Dempster Shafer Theory

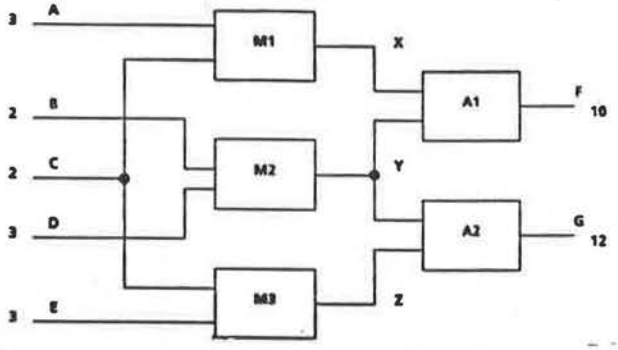
ATMS-based Implementations of full Dempster Shafer Theory have been done independently by Provan ([1988a], [1989]) and Laskey and Lehner [1988]. In addition, Pearl [1988], although he has not implemented a system, describes the semantic correspondence between the ATMS and DS theory in a manner almost identical to the ones presented by Provan and by Laskey and Lehner.

In describing these implementations, we need to introduce some ATMS terminology. The ATMS is a database management system which computes for a set Σ of propositional clauses a set of support (called a label) for each database literal in terms of assumptions, a distinguished subset of the database literals. The assumptions, which we denote by $\mathcal{A} = \{A_1, ..., A_l\}$, are the primitive data representation of the ATMS. The labels for literals thus summarise "proofs" in terms of a Boolean formula consisting of assumptions only. Hence an ATMS label is a restriction of the support set (defined earlier) to assumptions. The ATMS-based implementations assign mass only to assumptions. Additionally, for most problems the ATMS is restricted to Horn clauses, as it slows considerably with non-Horn clauses.

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 due to the introduction of new clauses. This is accomplished by storing the entire label set to avoid recomputing it every time it is needed.

A typical problem for which an ATMS is used is in the diagnosis of malfunctioning circuits, 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 4. Assumptions can be: (1) each component is working, where $\mathcal{N}_{A_i}$ signifies that

Figure 4: Circuit with faulty components



adder A_i is functioning normally, and $\mathcal{N}_{M_i}$ signifies that multiplier M_i is functioning normally; or (2) input data, e.g. $A = 3$, $B = 2$, etc. In the course of diagnosis, an assumption like $\mathcal{N}_{M_2}$, i.e. "M_2 is working", may be proved incorrect. For the circuit in Figure 4, the output at F is 10 instead of 12, implying that at least one combination of M_1, M_2, M_3, A_1 and A_2 is 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 minimal 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. For the example just described, the two conflict sets are represented logically as $\neg(\mathcal{N}_{A_1} \wedge \mathcal{N}_{M_1} \wedge \mathcal{N}_{M_2})$ and $\neg(\mathcal{N}_{A_1} \wedge \mathcal{N}_{A_2} \wedge \mathcal{N}_{M_1} \wedge \mathcal{N}_{M_3})$. Hence, the malfunctioning of the circuit shown in Figure 4 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.

It is immediately obvious that the ATMS can be used to compute the symbolic representation of Belief functions as described earlier. We given a brief description of the algorithm, and refer the reader to the relevant papers (Provan [1988a] and Laskey and Lehner [1988]). One drawback of the ATMS is that it computes support sets for literals only. Hence, Belief measures can be computed only for propositions which are literals.

ATMS-based Belief Computation Algorithm

Consider the process of computing $Bel(x)$ for some literal x. The necessary steps are:

1. Compute a DNF Boolean expression from the label set for x: if the label set consists of l separate labels, $\mathcal{L}(x,\Sigma) = \{\mathcal{L}_1(x),...,\mathcal{L}_l(x)\}$, this is given by
$$\mathcal{L}(x,\Sigma) = \bigvee_{\mathcal{L}_i \in \mathcal{L}} \bigwedge_{A_j \in \mathcal{L}_i} A_j.$$
2. Account for nogoods: a contradiction-free Boolean expression required is: $\mathcal{L}(x,\Sigma) \cap (\neg\Phi)$.
3. Compute a disjoint Boolean expression $disj(\mathcal{L}(x,\Sigma) \cap (\neg\Phi))$.
4. Substitute mass functions for the A_i's to calculate $Bel(x)$.

Example 1:
Recall the example (without nogoods) from page 3. The set of clauses, represented both as implications and in traditional clausal form, is

$A_1 \Rightarrow x_1$	$\overline{A_1} \vee x_1$
$A_2 \Rightarrow x_4$	$\overline{A_2} \vee x_4$
$x_1 \wedge A_3 \Rightarrow x_2$	$\overline{x_1} \vee \overline{A_3} \vee x_2$
$x_2 \wedge A_4 \Rightarrow x_3$	$\overline{x_2} \vee \overline{A_4} \vee x_3$
$x_1 \wedge A_5 \Rightarrow x_4$	$\overline{x_1} \vee \overline{A_5} \vee x_4$
$x_4 \wedge A_6 \Rightarrow x_5$	$\overline{x_4} \vee \overline{A_6} \vee x_5$
$x_2 \wedge x_4 \wedge A_7 \Rightarrow x_5$	$\overline{x_2} \vee \overline{x_4} \vee \overline{A_7} \vee x_5$

The masses assigned to the assumptions are:

ASSUMPTION	MASS
A_1	1.0
A_2	.8
A_3	.5
A_4	.7
A_5	.8
A_6	.6
A_7	.9
A_8	.4

The label sets the ATMS assigns to the literals are:

LITERAL	LABEL SET
x_1	$\{A_1\}$
x_2	$\{A_1, A_3\}$
x_3	$\{A_1, A_3, A_4\}$
x_4	$\{\{A_2\}, \{A_1, A_5\}\}$
x_5	$\{\{A_2, A_6\}, \{A_1, A_5, A_6\}, \{A_1, A_2, A_3, A_7\}, \{A_1, A_3, A_5, A_7\}\}$

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

$$\begin{aligned} Bel(x_5) &= \varrho\Big(\{\{A_2, A_6\}, \{A_1, A_5, A_6\}\{A_1, A_2, A_7, A_3\}, \{A_1, A_3, A_5, A_6\}\}\Big) \\ &= \varrho((A_2 \wedge A_6) \vee (A_1 \wedge A_5 \wedge A_6) \vee (A_1 \wedge A_2 \wedge A_7 \wedge A_3) \vee (A_1 \wedge A_3 \wedge A_5 \wedge A_6)) \\ &= \varrho(A_2)\varrho(A_6) + \varrho(A_1)\varrho(A_5)\varrho(A_6) - \varrho(A_1)\varrho(A_2)\varrho(A_5)\varrho(A_6) + \\ &\quad \varrho(A_1)\varrho(A_2)\varrho(A_3)\varrho(A_7) + \varrho(A_1)\varrho(A_5)\varrho(A_3)\varrho(A_7) - \varrho(A_1)\varrho(A_2)\varrho(A_3)\varrho(A_5)\varrho(A_7) - \\ &\quad \varrho(A_1)\varrho(A_2)\varrho(A_3)\varrho(A_6)\varrho(A_7) - \varrho(A_1)\varrho(A_3)\varrho(A_5)\varrho(A_6)\varrho(A_7) + \\ &\quad \varrho(A_1)\varrho(A_2)\varrho(A_3)\varrho(A_5)\varrho(A_6)\varrho(A_7) \\ &= 0.746. \end{aligned}$$

The Belief assigned to the literals is:

LITERAL	BELIEF
x_2	.5
x_3	.35
x_4	.96
x_5	.75

As mentioned earlier, the ATMS can dynamically update the label sets assigned to literals following the introduction of new clauses. This means that the Belief assignments to literals can also be dynamically updated.

In a logical framework, the label set for a set x of literals can be incrementally updated by support clause updating. For example, if the database is updated by a clause $x_5 \wedge x_7 \Rightarrow x_8$ such that x_5 and x_7 have already been assigned label sets and x_8 has not, then the label set for x_8 can be computed from the label sets for x_5 and x_7 as follows. If x_5 and x_7 have label sets $\{\{x_1, x_2\}, \{x_2, x_3\}\}$ and $\{\{x_1\}, \{x_4, x_6\}\}$ respectively, then x_8 is assigned the label set $\{\{x_1, x_2\}, \{x_2, x_3, x_4, x_6\}\}$ by taking a combination of the label sets for x_5 and x_7. Support clause updating is equivalent to pooling evidence for the antecedents of determine the *Belief* assigned to the consequent.

Example 2:

Consider the introduction of a new clause $x_2 \wedge A_8 \Rightarrow \overline{x_6}$, such that $\varrho(A_8) = 0.4$. Suppose we are given the information that x_4 and $\overline{x_6}$ are contradictory, so that a nogood Φ is formed:

$$\begin{aligned} \mathcal{L}(\Phi) &= \mathcal{L}(x_4) \wedge \mathcal{L}(\overline{x_6}) \\ &= \{\{A_2\}, \{A_1, A_5\}\} \wedge \{A_1, A_3, A_8\} \\ &= \{\{A_1, A_2, A_3, A_8\}, \{A_1, A_3, A_5, A_8\}\} \end{aligned}$$

The new assignment of Belief to literals is:

LITERAL	BELIEF
Φ	.192
x_2	.5
x_3	.41
x_4	.96
x_5	.57

6.3 Hypertree Implementations

There are several implementations of DS Theory based on hypertree embedding. These include DELIEF [Zarley *et al.*, 1988] and AUDITOR'S ASSISTANT [Shafer *et al.*, 1988]. These systems rely (for efficiency purposes and to ensure correctness) on determining a hypergraph embedding that can be arranged as a Markov tree. Belief functions are combined locally and then propagated through the Markov tree. These computations are similar to those taking place in implementations of Influence Diagrams (e.g. [Shachter, 1986]).

Other implementations of DS Theory are based on tree structure restrictions. Hierarchical evidence is one example of such a tree structure. Using this model, Gordon and Shortliffe [1985] propose an algorithm which computes approximate Belief functions, and Shafer and Logan [1987] implement a system which derive exact Belief functions. Hierarchical evidence enables a partitioning of Θ, and a great resulting efficiency over the unrestricted domain of size $|\Theta|$. We note that the Shafer and Logan algorithm is $O(nf)$, where $n = |\Theta|$ and f is the branching factor for the tree structure. This contrasts with the #P-completeness of Dempster's Rule.

7 RELATED WORK

DS theory is an active research area, as the following related literature shows. The work described in this paper is built on the work of several people, including d'Ambrosio [1987], Laskey and Lehner [1988] and Pearl [1988]. This paper formalizes the work of d'Ambrosio [1987] and Laskey and Lehner [1988]. They discussed the logical interpretation of DS theory with respect to the ATMS, whereas we describe it with respect to with respect to a more general framework, that of propositional logic. Thus this formalization can be used in any implementation based on propositional logic. In addition, neither d'Ambrosio nor Laskey and Lehner explicitly mentioned methods for computing the disjoint Boolean expressions necessary to compute Belief functions.

This paper extends several notions presented in Pearl [1988]. The notion of a Belief function summing proof paths has been formalized as the summing of minimal support sets. Pearl suggested the use of series-parallel reductions[23] to ensure disjointness. We explore many other Network Reliability algorithms, and summarize the complexity results for the problem of producing disjoint Boolean expressions.

[23]See the Inclusion/Exclusion methods presented in Appendix A.

Orponen [1989] has independently derived the #P-completeness of Dempster's rule. He uses a reduction from SATISFIABILITY, whereas we use a reduction from CONNECTEDNESS RELIABILITY.

Fagin and Halpern [1989] describe a general framework for the work presented here. Their work focuses on deriving a probabilistic interpretation for DS theory; here we focus on the propositional logic formulation. More specifically, Fagin and Halpern show DS Belief and Plausibility measures to correspond in a precise way to probabilistic inner and outer measures respectively. Given a probability measure μ, they consider a sample space $\mathcal{S}$[24] with non-measurable events, from which a probability space $(\mathcal{S}, \chi, \mu)$ is constructed.. They show that a probability structure defined on $(\mathcal{S}, \chi, \mu)$ is equivalent to a DS structure (and vice versa), provided that the domains considered are logical formulae rather than sets. In addition, they formally analyse the relationship between Nilsson's probabilistic logic and DS theory. Defining a Nilsson structure as a structure based on a set of weighted logical clauses, they show that every Nilsson structure is a DS structure, although the converse does not hold.

Ruspini [1987] has analysed DS theory in a manner which has many similarities to the analysis presented here. Ruspini frames DS theory within an epistemic modal logic which is equivalent to S5. He begins from a Carnapian analysis of the logical foundations of probability theory [Carnap, 1962]. Ruspini defines a possible world W_i as a mapping of a subset of atoms[25] $At = \{p, q, r, ...\}$ to $\{t, f\}$. $\mathcal{W}$ is the universe of possible worlds. A *frame of discernment* is the set of logical clauses built from At using $\{\vee, \wedge, \neg\}$. Ruspini distinguishes *objective* propositions (e.g. p, q) from *epistemic* ones (e.g. $\mathbf{K}p$, $\mathbf{K}q$). The epistemic propositions $\mathbf{K}p$, $\mathbf{K}q$) represent knowledge about their objective counterparts (p, q). For some subset $\mathcal{W}' \subset \mathcal{W}$ of possible worlds, Ruspini makes the following definitions:

$$\begin{aligned} \text{epistemic set:} \quad e(p) &= \{\mathcal{W}' \subset \mathcal{W} | (\mathbf{K}q \text{ is true}) \text{ iff } (p \Rightarrow q)\}, \text{ and} \\ \text{support set:} \quad k(p) &= \{\mathcal{W}' \subset \mathcal{W} | (\mathbf{K}p \text{ is true})\}, \text{ such that} \\ k(p) &= \bigcup_{q \Rightarrow p} e(q). \end{aligned}$$

It is easily observed that the support set $k(p)$ corresponds to the minimal support set $\xi(p, \Sigma)$ when $\xi(p, \Sigma)$ is restricted to singletons.

Ruspini then builds a σ-algebra such that measures are assigned to some set of subsets, rather than to every subset of the universe. He defines mass and support functions as follows, defining $P(q)$ to be the probability of q:

$$\begin{aligned} \text{mass function:} \quad m(p) &= P\{e(p)\}, \text{ and} \\ \text{support function:} \quad S(p) &= P\{k(p)\} \text{ such that} \\ S(p) &= \sum_{q \Rightarrow p} m(q). \end{aligned}$$

[24] This sample space is a classical probabilistic sample space, in which χ is a σ-algebra of subsets of $\mathcal{S}$; cf. [Fagin and Halpern, 1989] or [de Finetti, 1974].

[25] The atoms are propositional symbols in our terminology.

$S(p)$ corresponds precisely to a DS Belief function, provided that the mass function assigns a bpa only to singletons. $S(p)$ sums the mass assigned to the minimal support sets of p, with the support sets restricted to singletons.

Ruspini also describes the combination of knowledge in his formalism. If $\mathbf{K}p$ is true iff (1)$\mathbf{K}_1 p_1$ and $\mathbf{K}_2 p_2$ are true, and (2) $p_1 \wedge p_2 \Rightarrow p$, then we have:

$$\text{epistemic set combination:} \quad e(p) = \bigcup_{p_1 \wedge p_2 \Rightarrow p} [e_1(p_1) \cap e_2(p_2)], \text{ and}$$

$$\text{mass function combination:} \quad m(p) = \kappa \sum_{p_1 \wedge p_2 \Rightarrow p} P\{[e_1(p_1) \cap e_2(p_2)]\}, \text{ where}$$

κ is a normalization constant such that $\sum m(p) = 1$. This is a generalization of Dempster's rule, and requires an assumption of the independence of the epistemic algebras $\mathbf{K}_1$ and $\mathbf{K}_2$.

It is obvious that the mass function combination corresponds to the logical version of Dempster's rule presented here, with the summation being over the minimal support sets, i.e. over $\neg\xi(\Sigma_k, \Sigma) \Rightarrow \Sigma_k$.

We have shown that propositional logic is sufficient to formalize DS theory, and does so in a manner analogous to, and we argue more straightforwardly, than the modal logic proposed by Ruspini. In addition, we have shown how evidence combination can be described in terms of the well-known notion of the generation of prime implicates $\Pi(\Sigma)$, and from $\Pi(\Sigma)$ evaluating the minimal support sets.

8 DISCUSSION

We have described the relation between DS Theory and propositional logic. We have shown how the support clause $\xi(\Sigma_i, \Sigma)$ gives a notion of a symbolic explanation for Σ_i. In the same way, a symbolic representation for a DS Belief function provides a notion of a symbolic explanation, and the numeric value of the Belief can be viewed as a numeric summary of that explanation, or as the numerical assignment of the believability of the explanation. In addition, just as a logical model describes which propositions are true in a given world, the DS Belief assigned to a conjunction of focal propositions described the degree to which that set of propositions is true. Thus, to the extent to which logic and DS Theory overlap, DS Theory can acquire a logical semantics.

What does this analysis tell us about DS theory, and about the relations between DS theory and propositional logic?

1. Dempster's rule has been shown to be summing provability relations. It is primarily concerned with provability relations, and secondarily with manipulating uncertainty measures associated with those relations. *Belief* measures show the measure assigned to the necessity of a proof, and *Plausibility* measures show the measure assigned to the possibility of a proof.

This importance of provability accounts for what Pearl [1988] calls a "semantic clash" with probability theory. DS theory is a theory of uncertainty management complementary to probability theory, and it can be developed wholly independently of probability theory. In fact, the precise relation to probability theory has been shown [Fagin and Halpern, 1989]: *Belief* and *Plausibility* measures correspond to probabilistic inner and outer measures respectively.

However, even though there is a direct probabilistic interpretation of DS Belief and Plausibility measures, there is no direct probabilistic interpretation of Dempster's rule. We argue that this combination rule is the main divergence of DS theory from probability theory. It also provides intuition regarding when DS theory is appropriate and when it is not.[26] Ignoring normalization, Dempster's rule for *Belief* updating sums a set of mutually independent proofs for a proposition. This is quite different from the probabilistic notion in Bayes' rule of re-evaluating a measure based on new information. In fact, we have shown that Dempster's rule can be represented entirely in terms of well-known logical operations.

Hummel and Landy [1988] have described the relationships between DS theory and probability theory in terms of the statistics of the opinions of experts. A Belief value can be interpreted as the percentage of the set of experts who provide a Boolean vote for a particular opinion. From this perspective, Dempster's combination rule "contains nothing more than Bayes' formula applied to Boolean assertions, ... (and) tracks multiple opinions as opposed to a single probabilistic assessment." Thus Dempster's rule updates product sets of opinions instead of single opinions.

This interpretation of Dempster's rule is an alternative way of providing intuition into the differences between Dempster's rule and Bayes' rule. The common intuition in both the Hummel and Landy and our interpretation is that Dempster's rule is essentially operating using Boolean operations. Hummel and Landy refer to this rule in terms of Bayesian updating on Boolean opinions, such that the Boolean operations are fundamental.

2. DS theory, viewed in logical terms, generalises the logical notion of contradiction. Two arbitrary clauses can be defined (external to the logic) as being contradictory. This generalization is also present in truth maintenance [Doyle, 1979].

3. We have shown the close relationship between DS theory and Network Reliability. In Network Reliability, an event is defined as an assignment of functionality (i.e. functioning or non-functioning) to a set of components. Network Reliability is concerned with enumerating the existence in networks of events which define communication paths, and summing the probabilities of such disjoint events. A DS *Belief* measure enumerates the ways in which a proposition is provable (which can be represented as a proof *path*

[26] It is possible to create a different DS update rule, one which is tailored to specific situations, and which has a probabilistic interpretation. [Fagin and Halpern, 1989] speculate on this point as well.

through a graph), and sums the disjoint provability measures (i.e. paths). In fact, network reliability is DS theory restricted to Bayesian Belief functions.

4. This research provides insight into the relationship between DS theory and Nilsson's Probabilistic Logic. Leaving aside semantical notions, on one level the two theories are the same, as they assign a measure to a set of clauses, and then assign measures from this initial assignment based on the provability relations of the clauses. Both assign bounds to these provability measures.

   One major difference is in the approach to assigning bounds to the provability notions. Probabilistic logic uses a geometric approach based on linear programming. This method maps extreme vectors in the space of possible worlds **P** into extreme vectors in the space Π of probabilities of the sets of possible worlds. Using the fact that Π must lie within the convex hull of the extreme vectors of Π, a consistent region for Π can be defined.

   If Probabilistic Logic can be described as a model theoretic approach in which the extreme vectors of Π define a consistent convex hull, DS theory can be described as a proof theoretic approach in which the set of minimal proofs for a clause Σ_i are collected, and from the measure assigned to Σ, a measure is assigned to the existence of a proof for Σ_i. $Bel(\Sigma_i)$ can thus be seen as the *Prob* $\{\exists$ *a proof for proposition* $\Sigma_i\}$. If we describe both Probabilistic Logic and DS theory as consisting of a set of logical and consistent constraints, we can say that: in Probabilistic Logic Π (the probabilistic constraint) defines the consistent convex hull from which consistent worlds (i.e. the logical constraints) can be evaluated; in DS theory the logical constraints define a set of proofs from which the probabilistic constraints can be evaluated. Pearl [1988] describes this as follows: "Probabilistic logic ...(is) a set of hard (logical) restrictions imposed on a set of soft (probabilistic) models, while the DS theory ...(is) a set of soft restrictions imposed on a set of hard models."

This paper suggests several methods of implementing algorithms to compute DS Belief functions. For computing approximate DS Belief functions, randomized approximation algorithms are the most promising. For computing exact DS Belief functions, many implementations based on logical operations have been suggested. However, we argue that implementations based on hypertree embeddings and on certain Network Reliability approaches such as SDP will be more efficient than implementations which compute minimal support sets. Generating minimal support sets and then creating disjoint Boolean expressions from which to compute Belief functions does not exploit the structure of the problem. Hypertree embedding and certain network reliability approaches do exploit problem structure, and are inherently more efficient. However, when minimal support sets are required for other problem-solving purposes, then the support set-based computations are recommended. And, even though the support set-based implementations are inefficient, they do demonstrate the logical underpinnings of DS theory, and enable uncertainty reasoning to be applied in a formal manner to propositional rule networks.

Our future research includes further exploration of the differences among DS theory and other uncertainty reasoning formalisms, identification of appropriate applications of DS Theory and Probabilistic Logic implementations, and development of more efficient approximation algorithms for such computations.

ACKNOWLEDGEMENTS

I would like to thank Judea Pearl for helpful discussions, and Alex Kean for much help with the Logic-based DS Theory implementation. Thanks to the two referees for many helpful comments which have aided revision of this paper.

References

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

[Arnborg *et al.*, 1987] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of Finding Embeddings in a k-Tree. *SIAM J. Alg. Desc. Meth.*, 8:277–284, 1987.

[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. International Joint Conference on Artificial Intelligence*, pages 868–875, 1981.

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

[Berge, 1973] C. Berge. *Graphs and Hypergraphs.* North Holland, 1973.

[Carnap, 1962] R. Carnap. *Logical Foundations of Probability.* University of Chicago Press, Chicago, Ill., 1962.

[Cox, 1946] R. Cox. Probability, frequency and reasonable expectation. *American J. of Physics*, 14:1–, 1946.

[D'Ambrosio, 1987] 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.

[de Finetti, 1974] B. de Finetti. *Theory of Probability.* Wiley, NY, 1974.

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

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

[Dempster, 1968] A.P. Dempster. Upper and Lower Probability Bounds. *J. Royal Statistical Society*, 1968.

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

[Fagin and Halpern, 1989] R. Fagin and J. Halpern. Uncertainty, Belief and Probability. In *Proc. International Joint Conference on Artificial Intelligence*, pages 1161–1167, 1989.

[Fratta and Montanari, 1973] 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.

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

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

[Grosof, 1986] B. Grosof. An inequality paradigm for probabilistic knowledge. In L. Kanal and Lemmer J., editors, *Uncertainty in Artificial Intelligence*, pages 259–275. Elsevier Science Publishers B.V., North Holland, 1986.

[Howard and Matheson, 1981] R.A. Howard and J.E. Matheson. Influence diagrams. In R. Howard and J. Matheson, editors, *The Principles and Applications of Decision Analysis*, pages 720–762. Strategic Decisions Group, CA, 1981.

[Hummel and Landy, 1988] R.A. Hummel and M.S. Landy. A statistical viewpoint on the theory of evidence. *IEEE Trans. PAMI*, pages 235–247, 1988.

[Jerrum and Sinclair, 1988] M. Jerrum and A. Sinclair. Conductance and the Rapic Mixing Property for Markov Chains: the Approximation of the Permanent Resolved. In *Complexity of Computer Computations*, pages 235–244. Proc. STOC '88, 1988.

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

[Kong, 1986] A. Kong. *Multivariate Belief Functions and Belief Functions.* PhD thesis, Harvard University, 1986.

[Laskey and Lehner, 1988] K. Blackmond Laskey and P.E. Lehner. Belief Maintenance: An Integrated Approach to Uncertainty Management. In *Proceedings of the American Association for Artificial Intelligence*, pages 210–214, 1988.

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

[Orponen, 1989] P. Orponen. Dempster's Rule of Combination is #P-Complete. 1989.

[Pearl, 1986] J. Pearl. Fusion, Propagation, and Structuring in Belief Networks. *Artificial Intelligence Journal*, 29:241–288, 1986.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

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

[Provan and Ball, 1983] 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.

[Provan and Ball, 1984] J.S. Provan and M.O. Ball. Computing Network Reliability in time Polynomial in the Number of Cuts. *Operations Research*, 32:516–526, 1984.

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

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

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

[Provan, 1989] G. Provan. An Analysis of ATMS-based Techniques for Computing Dempster Shafer Belief Functions. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1989.

[Provan, 1990a] G. Provan. A Logic-based Analysis of Dempster Shafer Theory. *International Journal of Approximate Reasoning*, page to appear, 1990.

[Provan, 1990b] G. Provan. An Analysis of Exact and Approximation Algorithms for Dempster Shafer Theory. Technical Report in preparation, University of British Columbia, Department of Computer Science, 1990.

[Reiter and de Kleer, 1987] 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.

[Reiter, 1980] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence Journal*, 13:81–132, 1980.

[Ruspini, 1987] E. Ruspini. Epistemic Logics, Probability, and the Calculus of Evidence. In *Proc. International Joint Conference on Artificial Intelligence*, pages 924–931, 1987.

[Savage, 1954] L.J. Savage. *Foundations of Statistics.* John Wiley and Sons, 1954.

[Shachter, 1986] R. Shachter. Evaluating Influence Diagrams. *Operations Research*, 34:871–882, 1986.

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

[Shafer *et al.*, 1988] G. Shafer, P.P. Shenoy, and R.P Srivastava. AUDITOR'S ASSISTANT: A Knowledge Engineering Tool for Audit Decisions. Technical Report Working Paper No. 197, University of Kansas, School of Business, 1988.

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

[Shafer, 1982] G. Shafer. Belief Functions and Parametric Models. *J. Royal Statis. Soc*, B44:322–352, 1982.

[Shenoy and Shafer, 1988] P.P. Shenoy and G. Shafer. An Axiomatic Framework for Bayesian and Belief Function Propagation. In *Proc. AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 307–314. Morgan Kaufmann, 1988.

[Shenoy *et al.*, 1986] P.P. Shenoy, G. Shafer, and K. Mellouli. Propagation of Belief Functions: A Distributed Approach. In *Proc. AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 249–160, 1986.

[Smets, 1988] P. Smets. Belief Functions. In P. Smets, A. Mamdani, D. Dubois, and H. Prade, editors, *Non-Standard Logics for Automated Reasoning*, pages 253–286. Academic Press, 1988.

[Tarjan and Yannakakis, 1984] R. Tarjan and K. Yannakakis. Simple Linear-time Algorithms to Test Chordality of Graphs, Acyclicity of Hypergraphs, and Selectively Reduce Hypergraphs. *Siam J. Computing*, 13:566–579, 1984.

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

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

[Zarley *et al.*, 1988] D. Zarley, Y. Hsia, and G. Shafer. Evidential Reasoning using DELIEF. In *Proceedings of the American Association for Artificial Intelligence*, pages 205–209, 1988.

[Zhang, 1988] L. Zhang. Studies on Finding Hypertree Covers for Hypergraphs. Technical Report Working Paper No. 198, University of Kansas, School of Business, 1988.

[Zhuravlev, 1982] Y.I. Zhuravlev. Set-theoretical Methods in the Algebra of Logic. *Problemy Kibernetiki*, 8, 1982.

A NETWORK RELIABILITY TECHNIQUES

In this appendix we briefly review the major techniques for solving network reliability problems: Pathset/Cutset Enumeration and Pivotal Decomposition/Factoring.

A.1 Pathset/Cutset Enumeration

The path/cutset enumeration methods begin with the (minimal) set of paths/cutsets, and expand them so that they are disjoint. Two widely-used 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 $P(s,t)$ is given by

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

$\mathcal{P}_i$ is the event that all elements on the i^{th} minimal path-set are functioning. Enumerating the minimal cutsets of a graph is equivalent to enumerating the minimal paths, by Menger's Theorem (see [Berge, 1973]). We note that for any graph $\mathcal{G}(V,E)$, there are $2^{|E|-|V|+2}$ possible paths between any nonadjacent pair of nodes.

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

$$P(s,t) \;=\; 1 - P(\bigcup_{i=1}^{N} C^i_{s,t}), \tag{15}$$

where $C^i_{s,t}$ 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 $P(s,t)$ by path enumeration, each cutset must be disjoint. 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 more efficient.

Inclusion/exclusion (IE) Methods are based on the following simple expansion of parallel and series links:

- parallel links are computed using

$$\varrho(A_1 \wedge A_4) = \varrho(A_1)\varrho(A_4),$$

- and series links using

$$\varrho(A_1 \vee A_4) = \varrho(A_1) + \varrho(A_4) - \varrho(A_1)\varrho(A_4).$$

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

$$R(\mathcal{G}) = \sum_{i=1}^{s} \varrho(\mathcal{P}_i) - \sum_{i=1}^{s} \sum_{j<i} \varrho(\mathcal{P}_i \mathcal{P}_j) + \cdots + (-1)^{s-1} \varrho(\mathcal{P}_1 \mathcal{P}_2 \cdots \mathcal{P}_s). \quad (16)$$

An example of this enumeration technique is [Kim *et al.*, 1972]. As shown by equation 16, the terms alternate in sign, with the terms with $-$ signs being the double-counted terms.

Example:

We give an example of this method by computing the measure assiged to x_6 from the problem given on page 4.3. The non-disjoint measure assigned to x_6 is

$$\varrho_6 = \varrho_1\varrho_2 + \varrho_3\varrho_4 + \varrho_1\varrho_5\varrho_4.$$

Using the IE method, we will recursively create a disjoint expression by making the first two paths disjoint, to create expression R_2, and then making the entire expression disjoint by making R_2 and the third path disjoint.

$$\begin{aligned} \text{Set } R_1 &= \varrho_1\varrho_2. \\ \text{Then } R_2 &= R_1 + \varrho_3\varrho_4 - \varrho_1\varrho_2\varrho_3\varrho_4. \\ R_3 &= R_2 + \varrho_1\varrho_4\varrho_5 - \varrho_1\varrho_2\varrho_4\varrho_5 - \varrho_1\varrho_3\varrho_4\varrho_5 + \varrho_1\varrho_2\varrho_3\varrho_4\varrho_5. \end{aligned}$$

Hence, we obtain the disjoint measure assigned to x_6 as

$$\varrho_6 = \varrho_1\varrho_2 + \varrho_3\varrho_4 - + \varrho_1\varrho_4\varrho_5 - \varrho_1\varrho_2\varrho_3\varrho_4 - \varrho_1\varrho_2\varrho_4\varrho_5 - \varrho_1\varrho_3\varrho_4\varrho_5 + \varrho_1\varrho_2\varrho_3\varrho_4\varrho_5.$$

The **Sum of Disjoint Products (SDP)** method is based on the expanding all parallel paths using the following formula:

$$\varrho(\mathcal{P}_1 \vee \mathcal{P}_2) = \varrho(\mathcal{P}_1) + \varrho(\overline{\mathcal{P}_1} \wedge \mathcal{P}_2). \quad (17)$$

Thus, for a system with s paths, we obtain

$$R = \varrho(\mathcal{P}_1) + \varrho(\overline{\mathcal{P}_1}\mathcal{P}_2) + ... + \varrho(\overline{\mathcal{P}_1\mathcal{P}_2} \cdots \overline{\mathcal{P}_{s-1}}\mathcal{P}_s). \quad (18)$$

This methods 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 [1973], and then improved upon by Grnarov et. al [1979] and Abraham [1979]. The Abraham method has since been improved by Locks [1987] and by Beichelt and Spross [1987].

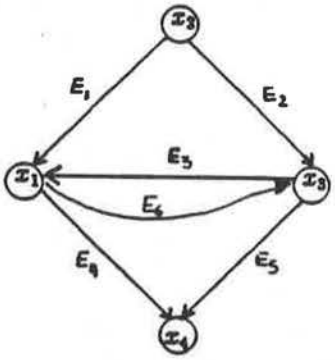
A.2 Pivotal Decomposition/Factoring

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 (disjoint) paths, such as that representing the bridge network shown in Figure 5. This theorem "factors out" edges in a graph by conditioning on such edges. Thus, you can condition on some edge e_j such that

$$P(s,t) = p_j\{P(s,t)\}_{p_j=1} + (1-p_j)\{P(s,t)\}_{p_j=0}, \quad (19)$$

where p_j is the failure probability of edge j, and $\{P(s,t)\}_{p_j=1}$ is the (s,t) reliability assuming edge j fails.

Figure 5: Bridge network: Example of a non-disjoint formula



Presence of edges E_3 and E_6 means the graph is not series-parallel decomposable.

B PROOFS

B.1 Logical Equivalence of DS Theory Proofs

Lemma 1 *All minimal support sets consist of assumptions only.*

Proof: Assume that some minimal support $\xi(x, \Sigma)$ contains a non-assumption literal x^*, i.e. $\xi(x, \Sigma) = \bigvee_i \overline{A_i} \vee \overline{x^*}$. By the definition of support set, if x^* occurs in $\xi(x, \Sigma)$, either x^* is not justified or $\xi(x, \Sigma)$ is not minimal. If x^* were justified, its antecedents would be in $\xi(x, \Sigma)$ instead of x^*. But all non-assumption literals are justified and $\xi(x, \Sigma)$ is minimal. Hence x^* cannot occur in $\xi(x, \Sigma)$. ■

Lemma 2 *The belief assigned to a proposition θ (which has corresponding logical clause Σ_k) can be computed from the support clause for Σ_k; i.e.*

$$Bel(\theta) = \varrho(\xi(x, \Sigma))$$

Proof: The Belief assigned to θ adds the measure of the subsets from which it is provable. The minimal support for Σ_k provides the minimal conjunction of assumptions from which it is provable, which is identical to the definition of Belief. ■

Lemma 3 *Dempster's rule for Belief combination, i.e.*

$$Bel(\theta) = \frac{\sum_{\bigcap_i \theta_i = \theta} Bel_1(\theta_1) Bel_2(\theta_2) \cdots Bel_m(\theta_m)}{1 - \sum_{\bigcap_i \theta_i = \emptyset} Bel_1(\theta_1) Bel_2(\theta_2) \cdots Bel_m(\theta_m)}, \tag{20}$$

corresponds to computing the measure assigned to Σ_j (the clause corresponding to θ, such that $Bel(\Sigma_j)$ is the measure assigned to a disjoint form of $\xi(\Sigma_j, \Sigma)$.

Proof: Each summand of the numerator of equation 20, $\cap_i \theta_i = \theta$, corresponds exactly to $\wedge_i A_i \Rightarrow \Sigma_j$, or $\neg\xi_k(\Sigma_j, \Sigma) \Rightarrow \Sigma_j$, where ξ_k is the k^{th} minimal support clause for Σ_j. The summation is thus over the minimal support sets of Σ_j. This summation is given by

$$\bigvee_{\xi_k \in \xi(\Sigma_j, \Sigma)} \xi_k(\Sigma_j, \Sigma).$$

The Belief assigned to this summation is given by

$$\begin{aligned} Bel(\Sigma_j) &= Bel(\bigvee_{\xi_k \in \xi(\Sigma_j, \Sigma)} \xi_k(\Sigma_j, \Sigma)) \\ &= \bigvee_{\xi'_k(\Sigma_j, \Sigma) \in disj(\xi(\Sigma_j, \Sigma))} \bigwedge_{A_i \in \xi'(\Sigma_j, \Sigma)} A_i. \end{aligned}$$

The denominator for equation 20 can be computed in an analogous manner. ■