Appl Intell
DOI 10.1007/s10489-010-0272-x

On the combination of logical and probabilistic models for information analysis

Jingsong Wang · John Byrnes · Marco Valtorta · Michael Huhns

Abstract Formal logical tools are able to provide some amount of reasoning support for information analysis, but are unable to represent uncertainty. Bayesian network tools represent probabilistic and causal information, but in the worst case scale as poorly as some formal logical systems and require specialized expertise to use effectively. We describe a framework for systems that incorporate the advantages of both Bayesian and logical systems. We define a formalism for the conversion of automatically generated natural deduction proof trees into Bayesian networks. We then demonstrate that the merging of such networks with domain-specific causal models forms a consistent Bayesian network with correct values for the formulas derived in the proof. In particular, we show that hard evidential updates in which the premises of a proof are found to be true force the conclusions of the proof to be true with probability one, regardless of any dependencies and prior probability values assumed for the causal model. We provide several examples that demonstrate the generality of the natural deduction system by using inference schemes not supportable directly in Horn clause logic. We compare our approach to other ones, including some that use non-standard logics.

Keywords Reasoning · Uncertainty · Probabilistic reasoning · Bayesian networks · Natural deduction proofs · Logic for knowledge representation

J. Wang · M. Valtorta (✉) · M. Huhns
Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA
e-mail: mgv@cse.sc.edu

J. Byrnes
SRI International, San Diego, CA, USA

Published online: 15 January 2011

1 Introduction

Support systems for information analysis must be able to quantify and track uncertainty in evidence findings, in data used by inferential processes, in the imperfect theories that emerge from the individual and collective experience of information analysts, and from other sources. Although they enjoy certain advantages in versatility and computational complexity, logical knowledge bases are ill-suited to represent uncertainty and then reason about it correctly, because knowledge representation languages based on classical logic do not provide facilities for representing and reasoning about uncertainty expressed in a probabilistic form. Bayesian probability theory defines the unique paradox-free method for reasoning with uncertainty. Recent research shows that, in principle, facilities for representing and reasoning about uncertain information can be provided by extending the logical framework to support such representations as multiple-entity Bayesian networks and probabilistic relational models, but the scalability of such approaches is questionable. We have been working on overcoming this problem in three ways. First, to simplify the construction and application of probabilistic models of situations of interest to an information analyst, we have implemented a simple version of Laskey and Mahoney's Bayesian network (BN) fragments approach [57]. A key feature of BN fragments is the distinction between nodes, which are in one-to-one correspondence with the nodes of a Bayesian network, and attributes of the nodes, which are used in the matching and composition process, as we described in previously published work [11, 98]. Second, we have developed an ontology of concepts that the designer of the decision support system can use in describing the nodes and attributes of Bayesian network fragments, which enables disciplined reuse and sharing of BN fragments [40]. Third, as we report

herein, we are developing methods that automatically convert logical proofs into Bayesian networks. A proof is derived (in our work, using a natural-deduction format) from the application of a logical knowledge base to a particular situation. The Bayesian network can then be used to reason about the uncertainty of data sources, the uncertainty associated with expert judgment, conflicting data, and conflicting judgments. Conflicting data will be a major issue as larger knowledge bases are used, and particularly as more of their content is extracted automatically from text, because most logic engines fail catastrophically upon encountering a contradiction.

The remainder of this paper is organized as follows. In Sect. 2, we introduce the logical and probabilistic models. We then analyze the complexity issues in Sect. 3. We describe the basic composition process in Sect. 4, discuss extensions in Sect. 5, and related work in Sect. 6. Then in Sect. 7 we provide the formal proofs of correctness of our approach. We briefly describe an early implementation in Sect. 8 and conclude this paper in Sect. 9.

2 Logical and probabilistic models

2.1 Two kinds of information

When we describe a situation that needs to be analyzed, we usually include two kinds of information. One part is about logic, where each statement expresses a direct and absolute relation among items of information, which can be used to infer new information. This kind of knowledge is best formalized in logic and includes

- Class-subclass statements, such as "dogs are mammals" and "coffee is a liquid"
- Part-whole statements, such as "intake valves are parts of cylinders"
- Definitional statements, such as "triangles have three sides" and "coffee is brown"
- Temporal statements, such as "3:00 p.m. occurs before 4:00 p.m."
- Spatial statements, such as "London is located in the UK"

Another part is about the probabilistic dependence of information. Examples of this kind of knowledge are statement such as:

- "Earthquakes cause burglar alarms to go off"
- "Terrorist cell X planned the bombing"
- "Suspect Y met with cell leader Z in London last March"
- "If the authors of a research paper work late at night, they drink coffee"

While the first kind of information can be viewed as a special case of the latter, where the allocated probability is one, it is convenient to represent purely logical relationships without the machinery of probability. Conversely, it is very difficult to represent expertise, especially as it relates to associational and causal information, without using probabilities [78, 79].

We could build the model for a whole situation from two parts too. The first part is the logic model, which translates logical statements directly into first-order logic formulas. In fact, due to its expressive power, logic is widely used in building knowledge bases. The second part is the probabilistic model in which we could use Bayesian networks to describe probabilistic dependency and probability distributions.

In a practical application, it is convenient to distinguish the description of a situation from findings and queries. When a query only involves logical information, we can get the answer using a basic inference mechanism. However, when a query involving uncertainty is placed, we will not be satisfied with an answer only coming from the logical model, since it is not based on the complete knowledge we have. For example, almost all diagnostic queries involve abduction and are of this kind [79].

We also want to support the discovery of changes in the probability of events that are not part of the original probabilistic causal model, but that become related to parts of the causal model via relationships captured in the logical model. So, it is necessary to explore ways of integrating the two kinds of information (logical and probabilistic) into one single model capable of doing reasoning with uncertainty. We use the methodology of representing the logical information in first-order logic, using natural deduction, translating natural deduction proofs to Bayesian networks, and composing the resulting Bayesian networks with already existing ones that directly capture associational and causal information. In this way, we integrate logical knowledge and probabilistic knowledge into a probabilistic model.

2.2 Alternative approaches

Our objective is to produce models of systems and situations that will be sufficiently accurate that they can be used—where appropriate—to predict future states, to understand operations, to illuminate the factors relevant to decisions, and to control behaviors. We have realized that some knowledge is more easily and naturally represented in the form of statements in a logic language and some is more naturally represented in a Bayesian network formalism. We would like to take advantage of the strengths of each formalism while combining them into a single coherent system. However, there are tradeoffs in how the two are combined. The tradeoffs are as follows:

- First, we can extend a logic formalism (in this case a natural-deduction proof system) to include causality.

This can be done by using special statements with associated conditional probabilities, for example, "coffee keeps me awake"

$$\text{coffee} \mid\!\rightarrow \text{awake}, \text{ where } P(\text{awake} \mid \text{coffee}) = 0.8 \quad (1)$$

The problem is that if there are several statements about the causes for the same concept (e.g., tea also keeps me awake), then the representation may mislead a modeler into assuming that it is possible to specify the whole conditional probability of the effect given the causes by providing only marginal conditional probabilities, without requiring assumptions such as independence of causal influence. In other words, it is difficult to get the probabilities correct, because each parameter in the special formalism just described (with the $\mid\!\rightarrow$ symbol) represents only a partial (marginal) specification of a large conditional distribution, which is not specified, and for which the number of independent parameters is (approximately) the same as the number of configurations of the possible causes.

– Second, we can try to include logic statements directly within a Bayesian network. This is problematic in the case of a large theory, even in the propositional case, because it requires the modeler to reconstruct proofs, which are best carried out by an automated theorem prover. It is especially confusing for a probabilistic modeler to deal with proofs that go beyond what can be represented by simple rules (definite Horn clauses). A probabilistic modeler knows well that $P(A \rightarrow B) = m$ (which is equivalent to $P(\neg A \vee B) = m$) is *not* equivalent to $P(B = \text{true} | A = \text{true}) = m$, but might need help (from an automated system or a logical modeler) to carry through complicated proofs.

The two existing formalisms of natural deduction and Bayesian networks are the most intuitive and most widely already understood ways of capturing their form of knowledge. The alternatives described, and especially the one with the $\mid\!\rightarrow$ symbol, may be worth pursuing. However, they would require further work than has been done to date. For the above reasons, we choose to pursue an integrated approach in which models are constructed from logical and probabilistic specifications, rather than by adding features to one of the two approaches. The probabilistic model provides the base for the integration, while the natural deduction theorem prover is used to automatically extract important logical knowledge to complement the probabilistic model from a purely logical knowledge base, which in some cases could be extremely large.

3 Complexity of logical and probabilistic inference

Our major scientific hypothesis is that integration of proofs and Bayesian networks will provide the main advantages of a full integration of logical knowledge bases with Bayesian networks, while keeping computational complexity sufficiently low for practical use. We do not attempt to prove the claim in this paper, but provide a proof of concept for a system that achieves the integration, including some examples. Some parts of the system (most importantly, the program for converting natural deduction proofs into Bayesian networks, briefly described in Sect. 4.2.1) have been implemented, while the others have been designed.

Most of the related decisions problems are intractable in general, but they can all be solved in polynomial time on networks whose treewidth is bounded [14]. So we first look at the general related classes of complexity, and then show the bounds of treewidth for our Bayesian network-based representation approach.

3.1 Classes of complexity

Given a Boolean formula, one of the most important and extensively studied problems is to decide its satisfiability, i.e., to decide whether there exists an assignment of its variables that makes the formula evaluate to True. The Boolean formula is composed of Boolean variables, Boolean connectives (AND, OR or NOT), and parentheses. The problem of determining a propositional Boolean formula's satisfiability is called the Boolean Satisfiability Problem (SAT). SAT is known to be NP-complete.[1]

Bayesian networks allow an explicit graphical representation of the probabilistic conditional dependences and independences among events or concepts, which reduces the number of probability assessments needed. The simplest form of probabilistic inference of Bayesian networks is the calculation of probability $P(X = x_i)$, where X is a propositional variable, and x_i is one of its possible values, which in binary case are True and False. Following the notation and conclusions from [12, 14, 85], we have the following problem representations and classes of complexity:

– PIBNET (Probabilistic Inference using Belief Networks), to compute $P(X = x_i)$, is #P-complete.
– PIBNETD (a decision-problem version of PIBNET), to decide $P(X = x_i) > 0$, is NP-complete.
– D-PR (a decision-problem version of PIBNET), to decide $P(X = x_i) > p$, where p is a number between 0 and 1, is PP-complete. Note that PIBNETD is a special case of D-PR in which $p = 0$.

It is also well known that PP and #P are closely related. PP may be considered to be the decision version of #P. In particular, $\text{P}^{\#\text{P}} = \text{P}^{\text{PP}}$, which can be paraphrased as follows: a

[1] When both universal and existential quantification of variables is permitted in the formula, the satisfiability problem is called the Quantified Boolean Formula (QBF) problem. QBF is known to be PSPACE-complete.

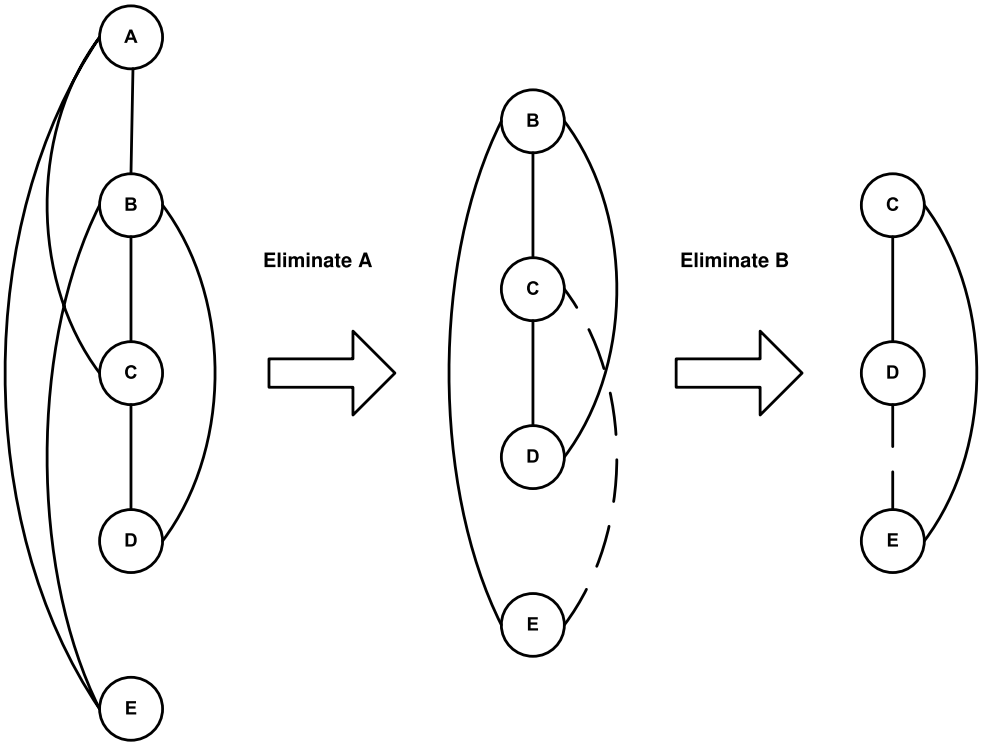

Fig. 1 Fill-in edges for elimination order $\langle A, B, C, D, E\rangle$ for the formula F

problem can be solved in polynomial time using an oracle in #P if and only if it can be solved in polynomial time using an oracle in PP [47].

It is well known that even approximating probabilistic inference is NP-hard [13, 77, 85]. It is also well known that the complexity of probabilistic inference is critically dependent on a graphical parameter called the treewidth of the Bayesian network, and more precisely that PIBNET, PIBNETD, and D-PR can be solved in time polynomial in the treewidth [5, 14, 84]. There is some evidence that most human-generated Bayesian networks have low treewidth.

3.2 Bounded treewidth

The treewidth of a graph G, $tw(G)$, is often characterized in terms of elimination orders. For a specific elimination order, its induced treewidth is the maximum neighborhood size in this elimination process. The treewidth of a graph is defined to be the minimum induced treewidth over all possible elimination orders.

Consider the propositional formula $F = (A \vee B \vee C) \wedge (\neg A \vee B \vee E) \wedge (\neg B \vee C \vee D) \wedge (\neg C)$. Figure 1 shows the partial process resulting from an elimination order $\langle A, B, C, D, E, F\rangle$ on the primal graph of F, where the dotted lines represent the edges added between each of the neighbors of the eliminated nodes. The primal graph of a formula in CNF form is a simple graph that has atomic formulas as vertices and has an edge for every two atomic formulas which occur in a common clause. The induced treewidth by this elimination order is 3. (Note that the treewidth of the primal graph of F is actually 2.) Figure 2 shows the translation of the formula into a Bayesian network; the family of node F is an AND table, while the families of nodes $C1, C2, C3, C4$ are OR tables. $C1, C2, C3$ and $C4$ represent clauses of F (e.g., $C1$ corresponding to the clause $(A \vee B \vee C)$). The first two steps of the variable elimination process of the moral graph of this Bayesian network are shown in Fig. 3 by the elimination order $\langle A, B, C, D, E, C1, C2, C3, C4, F\rangle$.

We now analyze the bounds of treewidth of a general formula.

Suppose F is a general formula in CNF form consisting of m clauses, represented by C_i, which totally contain n atomic formulas, represented by A_j, with $1 \leq i \leq m$ and $1 \leq j \leq n$. We denote the primal graph of F by G_p, and the corresponding Bayesian network by G_b. G_b is generated from F and has similar network structure as shown in Fig. 2. We assume G_p is a connected graph (if not, the isolated nodes will not affect the whole graph's treewidth, as their neighborhood sizes are always 0). We also assume G_p contains at least one node, i.e., $n \geq 1$.

3.2.1 Lower bound

Because node F is in a clique formed by itself and m clause nodes which are the only neighbors of node F, we can easily conclude that, in any elimination order, the maximum neighborhood size is at least m. So obviously $tw(G_b) \geq m$.

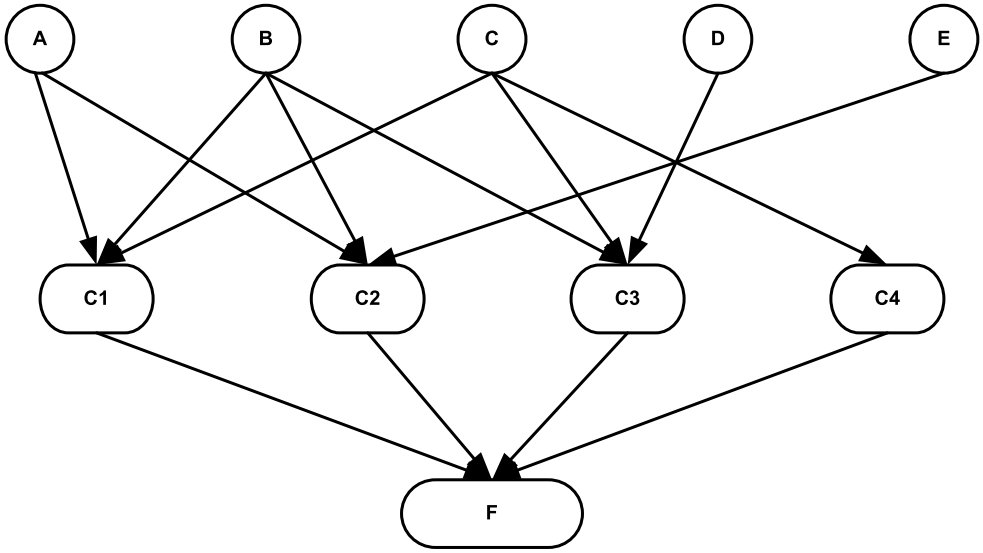

Fig. 2 Translation of the formula F into a Bayesian network [12]

3.2.2 Upper bound

Based on the definition of treewidth, we may use the maximum neighborhood size involved in any elimination order to measure the upper bound, as the treewidth should always be the minimum one. Also, because we want to analyze the change of treewidth from G_p to G_b, we intend to represent the bound of $tw(G_b)$ in terms of $tw(G_p)$.

So for G_b, we would start the elimination from the atom nodes at first, and particularly we choose the order of atom nodes which produces the real treewidth of G_p.

We represent this order by $\langle A_1, A_2, \ldots, A_n, C_1, C_2, \ldots, C_m, F\rangle$ for G_b. The nodes after A_n in this sequence come from the clause node set and F. Remember that $\langle A_1, A_2, \ldots, A_n\rangle$ produces the treewidth of G_p. Then for G_b, in the elimination process, the neighbors of any atom node to be eliminated will come from both the atom node set and the clause node set (the last atom node is an exception whose neighbors are all from clause node set). The neighborhood size for the atom node A_i can be denoted by $x_i + y_i$, $1 \leq i \leq n$, where x_i is the neighborhood size of this atom node in the elimination order $\langle A_1, A_2, \ldots, A_n\rangle$ of G_p, and y_i is the number of clause node neighbors connected to this atom node in G_b in this elimination process.

Because of the particular order we choose, we know $x_i \leq tw(G_p)$. Also, because there are totally m clause nodes, all of which exist till the last atom node is eliminated, we know $y_i \leq m$. So we conclude $x_i + y_i \leq tw(G_p) + m$.

For the elimination of clause nodes and node F, it is obvious that the maximum neighborhood size is m, as they form a $m + 1$ size clique.

So for this particular elimination order, the maximum neighborhood size is $\max\{x_i + y_i, m\}$. Because G_p is a connected graph with at least one node, $tw(G_p) \geq 0$. So $tw(G_p) + m \geq m$. So we can conclude $tw(G_b) \leq tw(G_p) + m$.

So, $m \leq tw(G_b) \leq tw(G_p) + m$.

3.2.3 Further discussion

In general, the probabilistic inference problem of Bayesian networks could be much harder than the satisfiability problem of the original logic theory. This conclusion seems to make the translation un-necessary. But this may not be the case when considering a special Bayesian network structure, like the one we shown in Fig. 2. We notice that one primal graph with a specific structure could have an exponential number of corresponding propositional theories, as a single primal graph has no restriction on the number of clauses the theory may contain. However, for any given theory, there is only one corresponding primal graph and we definitely know how many clauses it contains. Then based on the primal graph and the extra parameter m—the number of clauses in the theory, from the upper bound obtained above, we can see that the treewidth of the translated BN is not greater than the primal graph's treewidth plus m. So the actual resulting BN's complexity is much lower than in the general case. In addition, our approach is based on natural deduction proofs. The proofs typically involve only a small fraction of the logical knowledge contained in available knowledge bases, such as SUMO [72]. So usually[2] the resulting BN will only be a subgraph of the graph similar to Fig. 2. This lowers the actual complexity even further.

[2] Our conversion of natural deduction proofs introduces context nodes that are not present in the original logical knowledge base. These nodes have no children and, in common reasoning modes, are never instantiated. Therefore, they are barren nodes that do not affect the complexity of reasoning [46].

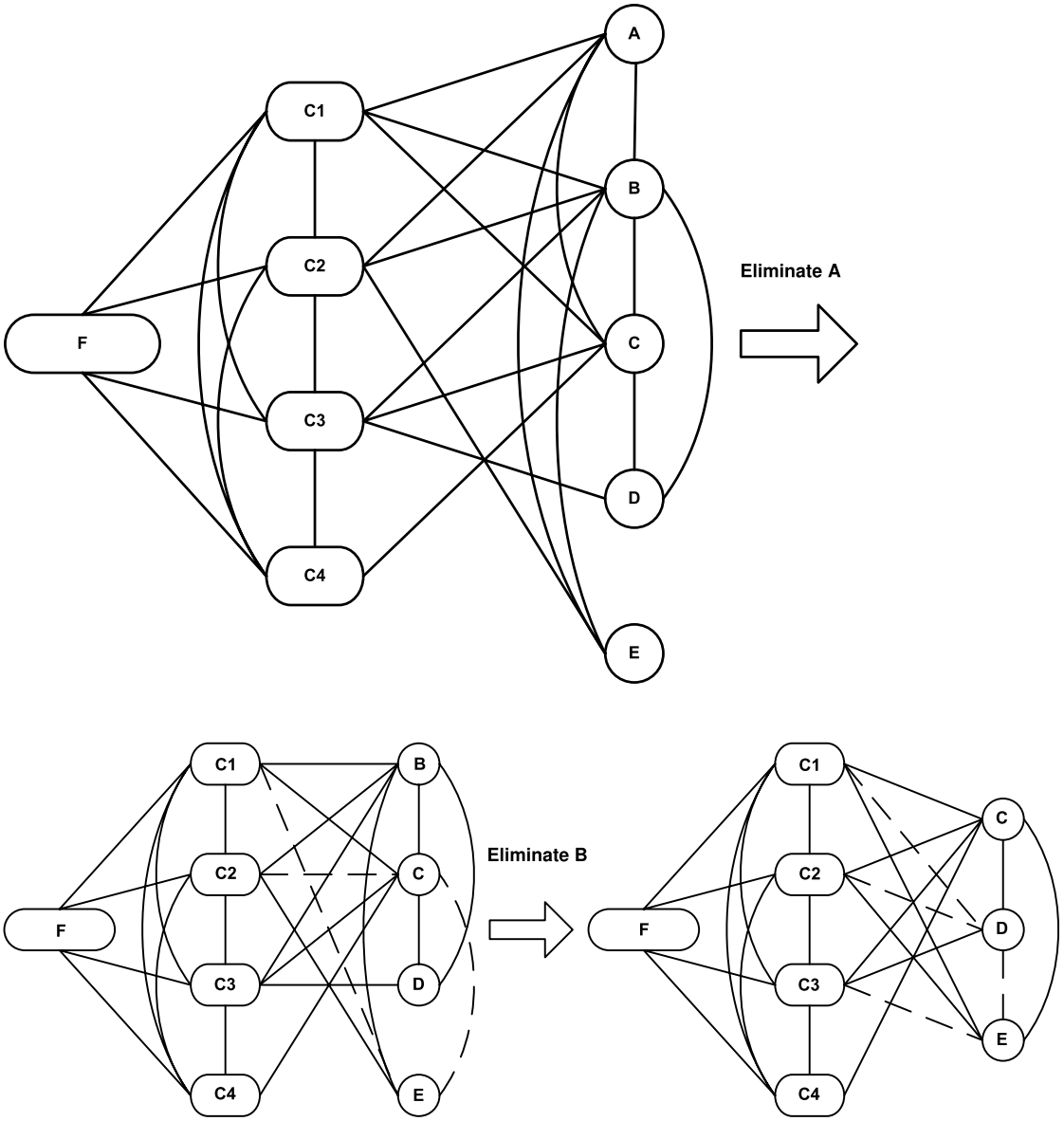

Fig. 3 The first and second steps of the solution of the satisfiability problem by variable elimination on the moral graph of the Bayesian network in Fig. 2

4 The process of composing

In order to describe the process of composing, we need to distinguish the variables represented in a Bayesian network from the variables used in a logic formula. The first kind of variables are in one-to-one correspondence with the nodes in a Bayesian network. Authors differ in the name used for such variables. For example, Neapolitan [69] uses the term propositional variable, Jensen and Nielsen [46] simply use the term variable, and Laskey [56] uses the term random variable. In every case, such variables partition the outcome space of interest. We trust that the context makes it clear which of the two kinds of variables we refer to. If necessary,

$$\dfrac{C \vee T \qquad \dfrac{C \to B \qquad C^{(1)}}{B} \qquad \dfrac{T \to B \qquad T^{(2)}}{B}}{B}\ (1,2)$$

Fig. 4 A natural deduction proof for brown liquids

we use *random variables* for the variables in one-to-one correspondence with the nodes of a Bayesian network and *logical variables* for the variables used in logic formulas. In some extensions of Bayesian networks, a random variable may correspond to a formula.

$$\dfrac{\Gamma \vdash C \vee T \qquad \dfrac{\Gamma \vdash C \to B \qquad \Gamma, C \vdash C}{\Gamma, C \vdash B}\to\text{-Elim} \qquad \dfrac{\Gamma \vdash T \to B \qquad \Gamma, T \vdash T}{\Gamma, T \vdash B}\to\text{-Elim}}{\Gamma \vdash B}\vee\text{-Elim}$$

Fig. 5 This proof tree makes contexts explicit. Γ stands for the set of assumptions $\{C \to B, T \to B, C \vee T\}$, and Γ, A stands for $\Gamma \cup \{A\}$

4.1 The basic process

Before composition, we suppose that there is a logical knowledge base (KB) consisting of logical rules and a Bayesian network (BN). Regarding events or propositions, KB describes their logical relations, while BN represents their probabilistic dependencies. We can simplify the process of composing by the following steps:

1. Natural Deduction
   We set a goal and then derive its natural deduction proofs based on the logical knowledge available in the KB.
2. Conversion
   We convert each proof into a separate Bayesian network.
3. Composition
   We compose the set of Bayesian networks obtained in the previous step with the original Bayesian network.

We use two propositional examples to illustrate the process. Note that, in this paper, the original Bayesian network is also called probabilistic casual model or probabilistic model. Correspondingly, the intermediate Bayesian networks resulting from natural deductions are called logical models or proof models. We assume that there are only atomic nodes (which represents atomic formulas) in the probabilistic model.

4.2 Two detailed propositional examples

4.2.1 Brown liquids

We provide an example of using the integrated logical and probabilistic reasoning system. Since the propositional theory that formalizes the example includes at least one non-Horn clause, i.e., at least one clause that includes two non-negative literals, the theory cannot be handled correctly by Horn clause logic[3] or by forward chaining rule-based systems such as JESS or CLIPS. The example formalizes the following story: my cup contains either coffee (C) or tea (T). Coffee is a brown liquid (B). Tea is a brown liquid. Thus it can be concluded that my cup contains a brown liquid.

The axioms in the knowledge base that formalize the story are:

[3]Pure Prolog can be considered a Horn clause logic representation language; however, Prolog is a general programming language that can be used to implement complicated reasoning engines.

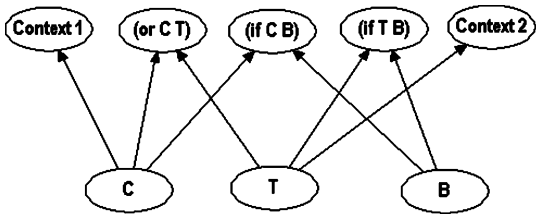

Fig. 6 The Bayesian network representation of the brown liquids proof

English assertion	Logical representation
My cup contains either coffee or tea	$C \vee T$
Coffee is a brown liquid	$C \to B$
Tea is a brown liquid	$T \to B$

We want to show B. Note that the theory allows for both tea and coffee to be in my cup at the same time. A natural deduction proof for B is given in Fig. 4. The proof consists of three steps: two $\to$-elimination steps and one $\vee$-elimination step. The $\vee$-elimination step, which corresponds to a case analysis step, requires us to prove B from the assumption C and separately from the assumption T. The assumptions are indicated by superscripts and discharged at the $\vee$-elimination step. These assumptions are used in the proof of the $\to$-elimination steps.

Figure 5 presents the same proof as Fig. 4, but in a way that emphasizes the contexts used. The proof in Fig. 5, which includes the $\vdash$ symbol, will remind some readers of the sequent calculus. However, it is directly a natural deduction proof with the exact same structure as the proof in Fig. 4; it only uses a different syntax to denote active assumptions.

The natural deduction proof is converted to a Bayesian network in the following way. Each non-atomic formula used in the proof is the child of its component subformulas, with a conditional probability table (CPT) that encodes the main connective introduced or eliminated. For example, in Fig. 6, the (nodes corresponding to the) atomic formulas C and T are parents of the (node corresponding to the) formula (or C T), and the CPT for the family of those three nodes, $P((\text{or } C\ T) \mid C,\ T)$ is an OR table. The Bayesian network also represents the nonempty contexts (sets of assumptions) used in the proof. For example, formula C is the context for the first step of the proof, namely the implication elimination with premises C and (if C B) and conclusion B. Accordingly, the node corresponding to formula C is a parent of the node *Context*1 in the Bayesian network. In the

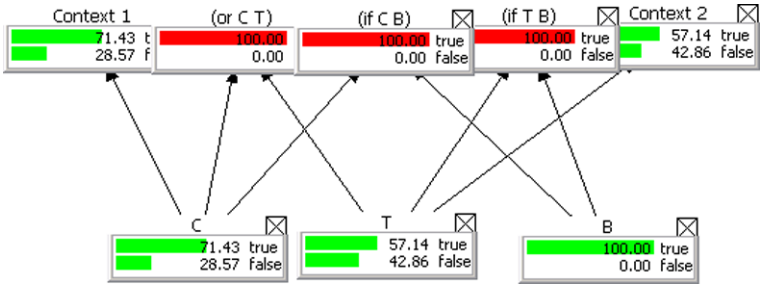

Fig. 7 B logically follows from the axioms in the brown liquids domain

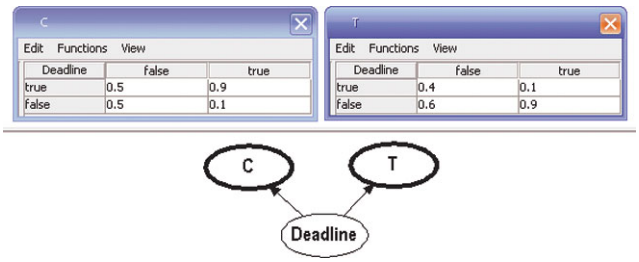

Fig. 8 A probabilistic causal model that relates work deadlines to coffee and tea in my cup

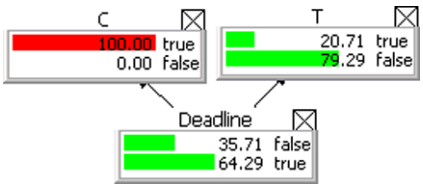

Fig. 9 The probability update conditioned on *C*

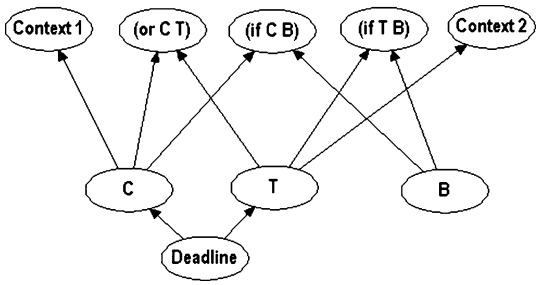

Fig. 10 A model composed from logical and probabilistic components

CPT for a context node, the context is true if and only if all of its parents are true. As an illustration, consider the Bayesian network structure of Fig. 6.

The construction algorithm just outlined ensures that any possible (i.e., non-zero probability) configuration (i.e., assignment of truth or false values) of the variables in the Bayesian network that correspond to formulas is a true interpretation (a model) of the formulas that appear in the steps of the proof and that no other assignments have positive probability, when the value true is entered as evidence for the (nodes corresponding to the) formulas of the theory. Figure 7 illustrates this, where it is shown that the only state of positive probability of the *B* variable is the one in which *B* is true, when evidence is entered for (if *T B*), (if *C B*), and (or *C T*). The probabilities shown in Fig. 7 are computed using the commercial Bayesian network shell Hugin (www.hugin.com), which uses the junction tree algorithm for probabilistic inference [46]. The evidence entered is indicated by red bars in a color version of the figure. Moreover, for a particular set of contexts, the possible configurations are models of the assumptions in the contexts and of the formulas.

Now, imagine that we have probabilistic information relating some of the variables in our domain of interest. In particular, following our example, imagine a probabilistic causal model is available that relates the presence of tea or coffee in my cup to the amount of work I need to get done before the end of the workday, as described in Fig. 8 using Hugin. Figure 9 shows one example of probability update of this casual model conditioned on *C*. We can now compose the logically derived model of Fig. 6 and the probabilistic causal model of Fig. 8 into a single model using the Bayesian network fragment composition algorithm described in [11] and obtain the combined model of Fig. 10.

The combined model is a Bayesian network and can be subjected to processing as any such network. The most important kind of processing is to compute the posterior probability of each variable in the network given a set of findings (i.e., evidence). For example, we may be interested in the probability of a deadline given that we observe coffee in my cup and that some axioms hold (to meet deadlines we work late and consume coffee to stay awake). The posterior probabilities are shown in Fig. 11, where we observe a roughly 68% probability of my working on a deadline, which happens to be a bit higher than the one got from the single probabilistic model as shown in Fig. 9, because of the introduction of logical relationships.

4.2.2 *Swimming pool*

The second example formalizes the following story: I have a swimming pool (A). If I have a swimming pool and it does not rain (D), I will go swimming (B). If I go swimming, I will get wet (C). If it rains, I will get wet. It can thus be concluded that I will get wet. Loveland and Stickel [60] use this

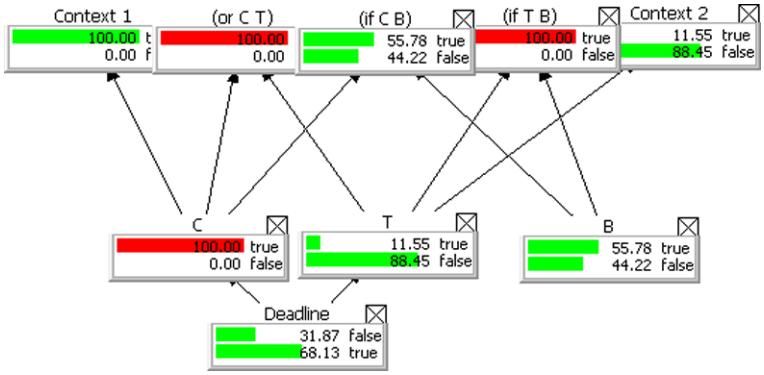

Fig. 11 Probability update in the model of the previous figure

$$\dfrac{\dfrac{\dfrac{D \vee \neg D \qquad \dfrac{B \to C \qquad \dfrac{(A \wedge \neg D) \to B \qquad \dfrac{A \qquad \neg D^{(1)}}{A \wedge \neg D}}{B}}{C}}{C} \qquad \dfrac{D \to C \qquad D^{(2)}}{C}}{C}\ (1,2)$$

Fig. 12 A natural deduction proof for the swimming pool example

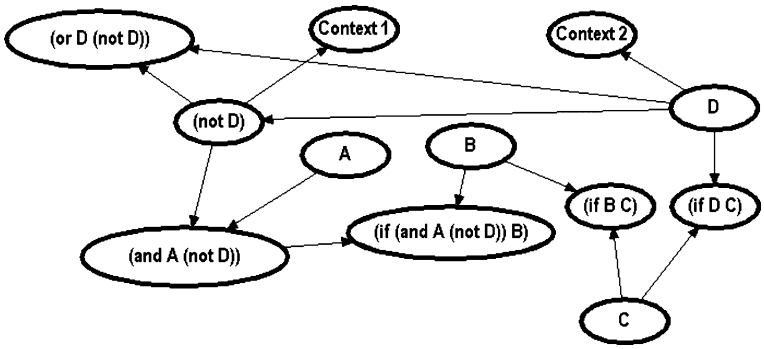

Fig. 13 The Bayesian network representation of the swimming pool proof

example to show that goal trees are incomplete and to motivate the use of ancestor contradiction checks, which they show to be complete.

The axioms in the knowledge base that formalizes the story are:

English assertion	Logical representation
I have a swimming pool	A
If I have a swimming pool and it does not rain, I will go swimming	$(A \wedge \neg D) \to B$
If I go swimming, I will get wet	$B \to C$
If it rains, I will get wet	$D \to C$

We want to show C. A natural deduction proof for C is given in Fig. 12, where we omitted the proof of $D \vee \neg D$. This proof consists of five steps: one $\wedge$-introduction step, three $\to$-elimination steps, and one $\vee$-elimination step. The $\wedge$-introduction step and the last $\to$-elimination step require one assumption each, namely $\neg D$ and D. The $\vee$-elimination step, which corresponds to a case analysis step, discharges both assumptions.

The Bayesian network representation of the proof is given in Fig. 13. We adopt the common convention that $\neg D$ is a shorthand for $D \to \bot$. For simplicity, we omitted the node representing $\bot$ and the edge from it to $\neg D$. Figure 14 shows the configuration of the Bayesian network variables when the evidence indicating that the four axioms are true is entered; note that C is true when the axioms are true. If we have a probabilistic model as in the previous example, we can do composition in a similar way.

5 Extensions

The previous section uses propositional examples to show the basic idea of composition, especially the important conversion step from natural deduction proofs to Bayesian networks. In solving real world problems, we are always facing more complex situations. The possible problems could be: How do we handle the multiple proof case? Where do goals come from? We will discuss these related problems in this section.

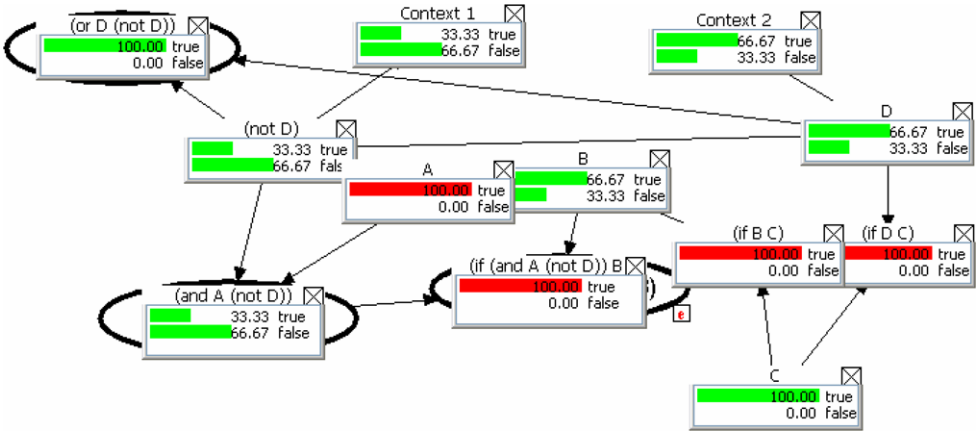

Fig. 14 C logically follows from the axioms in the swimming pool domain

5.1 Composing more proofs together

In both examples shown in the previous section, only one natural deduction proof is used for the composition with the existing probabilistic model. In fact, when more proofs are available, we can compose all of them with the probabilistic model, in order to get more complete knowledge for probabilistic reasoning.[4] From another point of view, we may think of the process of having multiple proofs composed together as the one in which we keep updating the probabilistic model for the next round of composition. In the initial round, we compose the original probabilistic model with one proof model, and then use the resulting composite model as the new probabilistic model to compose with another proof model in the next round. Some practical issues will be discussed near the end of this section.

5.2 Goal setting

There are two ways of generating the goals, based on which correspondingly we can categorize compositions into two types: Query Based Composition and Probabilistic Model Based Composition.

5.2.1 Query based composition

This type of composition occurs when the user submits a query to the system, i.e., the user has specific interest in some problem. In this case, we set this query or related formula[5] as a goal and get its natural deduction proofs. After converting each of them into a Bayesian network, the proof model, we start its composition with the probabilistic model if it exists. The resulting composite model can then be used to reason about the uncertainty of the query. This kind of composition can be considered as a special case of another type of composition shown in the following section, by regarding the query node as a simple probabilistic model or by extending the current probabilistic model to contain this query node.

5.2.2 Probabilistic model based composition

This type of composition is used mostly when the user does not have a specific question, but wants to augment the existing probabilistic model to support reasoning based on more complete knowledge. In this case, the natural deduction proofs are used to complement the original probabilistic model. There could be three scenarios regarding the way goals are set:

– Scenario one:
  In this scenario, we integrate all the possible proofs of each node[6] of the probabilistic model, as long as the proofs introduce no new atomic nodes, to the probabilistic model. Scenario one occurs when we are interested in reasoning over a set of nodes, the basic probabilistic dependencies of which have already been specified through the probabilistic model by domain experts. However, we want to exploit the logical knowledge base to capture more connections among these nodes and build a more accurate model for reasoning.
  Based on the definition of conversion of natural deduction proofs to Bayesian networks, all the newly added logical nodes from proof model, which are all compound

[4] Obviously, proof search is also complex. In practice, we should bound the search depth and time, since proofs might be infinite or we do not need that many proofs.

[5] "Related formula" means that its natural deduction contains the query formula. This will be explained more in the next section.

[6] Note that because both the probabilistic model and the proof model are standard Bayesian networks, we simply use "node" to call each random variable in the networks, which can represent atomic or compound formulas as shown in Fig. 6. Specifically, the node corresponding to an atomic formula is called atomic node, and the node corresponding to an compound formula is called compound node. The proof model generally contains both kinds of nodes. The probabilistic model is temporarily assumed to contain atomic nodes only.

nodes (because of the restriction on the proof that no new atomic nodes are introduced), will be children of the original nodes in the probabilistic model and therefore d-separate (as converging connections) the original nodes, when the logical nodes are not set as evidence findings. So adding them will not affect the probability distribution of the original probabilistic model. But after their instantiation, the logical nodes will open up communication among those pre-existing nodes. It is obvious that the more proofs of natural deduction integrated into the probabilistic model, as long as the restriction is not violated, the more dependencies of those pre-existing nodes of the probabilistic model will be revealed from the resulting composite model.

– Scenario two:
  In this scenario, we integrate all the possible proofs of each node of the probabilistic model. Here we relax the restriction to the proofs in scenario one, i.e. the proofs may introduce new atomic nodes. Scenario two occurs when we have further interest in the nodes outside the probabilistic model (e.g., there might be findings that are not described in the original probabilistic model) and additionally we still hope that the composite model could have reasonable size considering accuracy and efficiency. The restriction that only proofs of the nodes in the probabilistic model can be integrated ensures that the integrating process is finite. The same d-separations as in scenario one still exist, since all the logical nodes still serve as children of the original nodes. The new atomic nodes cannot communicate with the original probabilistic nodes if the logical nodes are not set to be findings, because they are also parents of the logical nodes (by the process of conversion).
  Scenario two is just an extension of scenario one. We have a bigger view of the world while the resulting model still remains affordable regarding building cost and size. Compared with scenario one and three, scenario two is the most used.
– Scenario three:
  In this scenario, we integrate all the possible proofs, which are not restricted to be only for the nodes in the probabilistic model. The proof can be for a new literal (i.e., there may be a proof for an atomic node or its negation that is not in the probabilistic model), as long as the converted proof model has common atomic nodes to link itself with the original probabilistic model. Scenario three is just an extension of scenario two. The set of proofs that we could integrate into the probabilistic model is still a countable set, but when the knowledge base is huge, searching could be very expensive.
  This kind of model composition is very helpful. Sometimes we know many nodes (nodes denote the entities' properties) existing in the world, e.g., we get some concepts from the existing knowledge base. But limited by our understanding capability, we can only build a smaller and simpler probabilistic model, which contains just some of these nodes. With the methodology described in the paper and the existing knowledge base, we could use natural deduction proofs to bring the logical relationships into the original probabilistic model, by integrating the proofs, consisting of the atomic formula nodes originally isolated from the probabilistic model and the compound formula nodes coming from logical rules, into the original probabilistic model. So we have used the logical knowledge to extend the original probabilistic model for more accurate uncertainty reasoning.

The brown liquids example of the previous section can be used for a simple illustration of the three scenarios. If we assume that the user submits a query on B at first, then the brown liquids example is a typical query based composition. From another perspective, if we assume that a probabilistic model is given at first, around which the composition is to be built, we can think of the same example as a probabilistic model based composition. Note that the composition in Fig. 10 is directly a use of scenario three. In this example, the probabilistic model is shown in Fig. 8. The proof model for B (shown in Fig. 6) can be composed with the probabilistic model because it has common nodes C and T. If we consider having a probabilistic model which is similar to the one shown in Fig. 8 but contains one extra node B, besides three pre-existing nodes *Deadline*, C, and T, then the composition with the same proof model in Fig. 6 could be regarded as a use of scenario one, as the proof is built for the node contained in the probabilistic model and no new atomic node is introduced. (Note that we ignore the context nodes.)

General composition algorithms We can use Fig. 15 to show different types of knowledge involved in composition. Here, P represents the atomic nodes from the probabilistic model. L represents the logical rules and findings available. Each finding might be an atom or its negation, i.e., a literal; compound formulas cannot be used as findings. O represents all the literals (i.e., each member of the set O is either an atomic node or the negation of an atomic node) contained in some formula of L, but not contained in P[7]. The area in the thin rectangle (purple and red in a color version of the figure) represents some of the finding nodes in L that are also in either P or O. For example, some findings are relevant to the probabilistic model, some are not. We group the finding nodes into L because they are combined with logic rules to make natural deduction proofs.

[7] If an atomic formula only appears negated, it is not necessary to set up the goal to be proved as the non-negated atom. A literal has the form A or $A \rightarrow \bot$, which is often conveniently represented as $\neg A$.

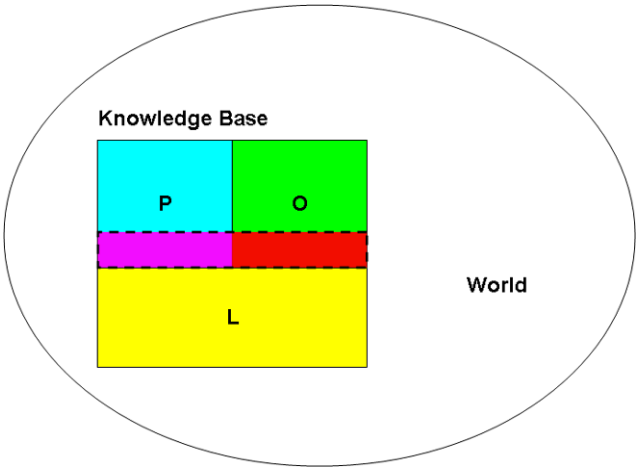

Fig. 15 Relationships among the types of knowledge: atomic nodes in the probabilistic model (P), logical rules (L), other nodes (O). P and O are disjoint. We are interested in finding nodes in L that are common to P and O

Based on this knowledge structure, we propose three algorithms for composing proof models with the probabilistic model, corresponding to each scenario.

Algorithm 1 Scenario One

Require: the probabilistic model BN, the set P of atomic nodes in BN, the set L of the logical nodes (including logical rules and findings)

for all e such that $e \in P$ **do**
  for all N such that N is a natural deduction proof of e, based on L **do**
    $Qualified \Leftarrow$ **true**
    for all a such that a is an atomic node of N **do**
      if $a \notin P$ **then**
        $Qualified \Leftarrow$ **false**
        break
      end if
    end for
    if *Qualified* is **true then**
      $BN = BN \oplus N$ {$\oplus$ means composition}
    end if
  end for
end for
return BN

When we build a composite model, we can follow the rule of cumulation. We start from scenario one and then see whether the current model satisfies our requirement. Based on scenario one, further, we can build a bigger model corresponding to scenario two. When we cannot retrieve enough valuable knowledge from scenario two, we will extend it to scenario three. So each scenario is based on its previous scenario. Especially for scenario three, we will not begin to add the proofs related to set O until we have finished composing the proofs based on set P.

Algorithm 2 Scenario Two

Require: the probabilistic model BN, the set P of atomic nodes in BN, the set L of the logical nodes (including logical rules and findings)

for all e such that $e \in P$ **do**
  for all N such that N is a natural deduction proof of e, based on L **do**
    $BN = BN \oplus N$ {$\oplus$ means composition}
  end for
end for
return BN

Algorithm 3 Scenario Three

Require: the probabilistic model BN, the set P of atomic nodes in BN, the set L of the logical nodes (including logical rules and findings - we consider only atomic findings), the set O representing all the literal nodes in L but not in P.

for all e such that $e \in P \cup O$ {elements of P should be tried before of O} **do**
  for all N such that N is a natural deduction proof of e, based on L **do**
    if $N \cap BN \neq \emptyset$ {$\cap$ means intersections of nodes} **then**
      $BN = BN \oplus N$ {$\oplus$ means composition}
    end if
  end for
end for
return BN

Examples We use an example to show how to follow the algorithms to make compositions. Formulas in the example do not have a real meaning and are just used to show the procedures.

First, we have a probabilistic model as shown in Fig. 16. In the knowledge base, we also have logical rules as shown in Table 1. Based on the probabilistic model and logical rules, we can generate:

$$P = \{A, B, C\},$$

$$L = \{A, A \to B, A \to C, C \to B, A \to D, D \to B, A \to E, B \to E, F \to G\}, \quad \text{and}$$

$$O = \{D, E, F, G\}.$$

– Example for scenario one: In scenario one all proofs involve only nodes that are in the probabilistic model. The proofs that are used for the example are given in

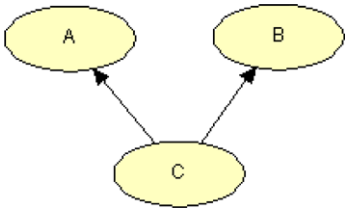

Fig. 16 The probabilistic model used for composition of scenarios 1, 2, and 3

Table 1 The logical rules for the examples illustrating the three composition scenarios

1	A
2	$A \rightarrow B$
3	$A \rightarrow C$
4	$A \rightarrow D$
5	$A \rightarrow E$
6	$B \rightarrow E$
7	$C \rightarrow B$
8	$D \rightarrow B$
9	$F \rightarrow G$

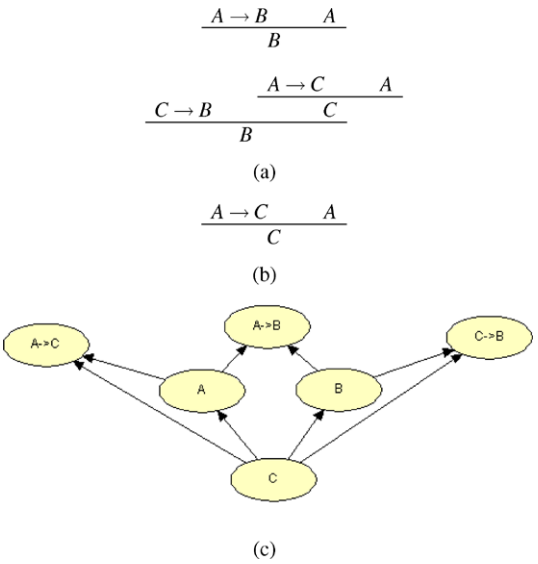

Fig. 17 Scenario one: (**a**) Natural deduction proofs of B, (**b**) a natural deduction proof for C, and (**c**) the output of the model composition

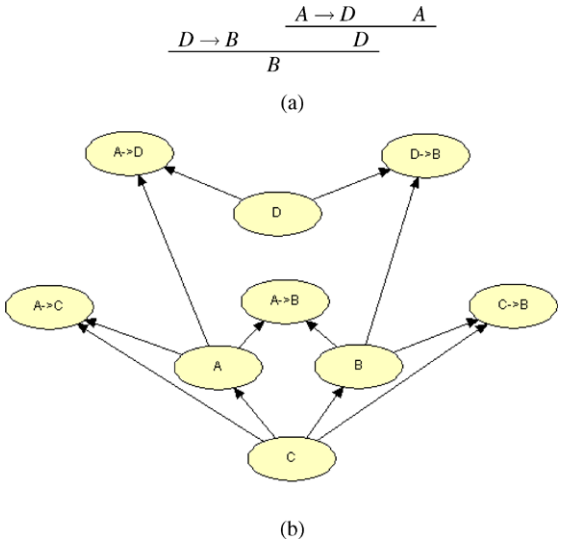

Fig. 18 Scenario two: (**a**) Another natural deduction proof of B and (**b**) the output of the model composition

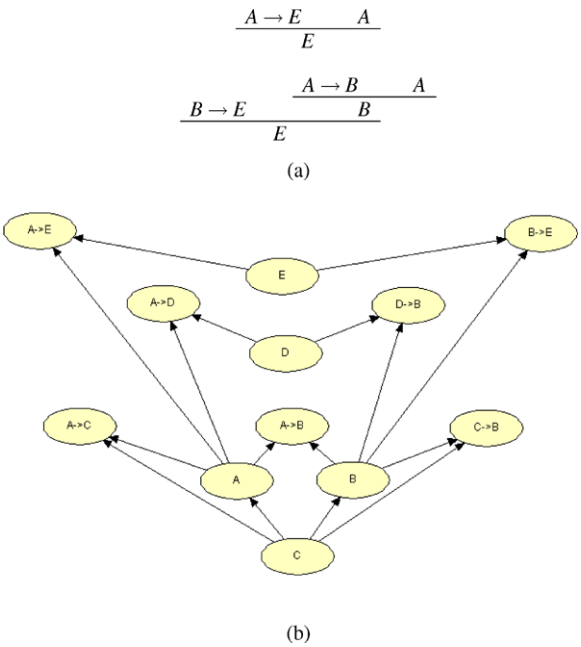

Fig. 19 Scenario three: (**a**) Natural deduction proofs of E in scenario three and (**b**) the output of model composition. Notice that we use B to prove E, and possibly there are different proofs of B that may then lead to different proofs of E. But, since we have added those proofs of B in previous scenarios (scenarios one and two), it is not necessary to list all the different proofs of E based on B in (**a**)

Fig. 17(a) and (b). The output of the model composition process is given in (c).

– Example for scenario two: In addition to the proof shown in scenario one, we add the proof of B in Fig. 18(a), which cannot be used in scenario one as it will introduce new atomic node D. The output of the model composition process is given in Fig. 18(b).
– Example for scenario three: In scenario three, a proof is introduced as long as there is an undirected path linking it to the probabilistic model. In addition to the proofs used in scenario two, we add the proof of E in Fig. 19(a). The output of the model composition process is given in Fig. 19(b).

5.3 Discussion

5.3.1 Adaptation

For a real world probabilistic model, it is possible that it contains not only atomic nodes but also compound nodes. So we need to change the composition algorithm to adapt to this situation. The solution is simple: just ignore the compound nodes of the original probabilistic model, i.e., only consider proofs of its atomic nodes in the composition process. This ensures that in one composition we only merge the roots of the proof model, which are atomic, with the probabilistic model.

When considering composition involving multiple proof models, the probabilistic model keeps changing after each composition step and we are always adding new proofs to the latest probabilistic model. In this case, the updated probabilistic model will start to contain compound nodes that do not exist in the original probabilistic model but come from the proof model used in the previous step. For such compound nodes in the next composition step, when we find a new proof model containing them, we do not need to add them to the probabilistic model once again. This is because, for any compound node used by different proofs, its parent nodes are always the same. This implies the same network structures and conditional probability distributions. Also, it is obvious that when different proofs of a given node are available, the corresponding natural deductions should have at least one logic rule used in one proof but not in the other. Therefore, the process of composing another proof is just adding to the current probabilistic model different nodes and edges. The existing nodes and edges brought by other previous proofs are unchanged. At this time, node mergers occur not only on the roots, but implicitly also on the existing compound nodes brought by previous proofs. Meanwhile, it is still the case that only children are added to the current probabilistic model, i.e. logical consistency still holds. When adding different proofs of different nodes, the process is the same. We never change the existing nodes. We just add new nodes that do not exist before in the current probabilistic model.

So we can make some changes to the algorithm by adding one step for the preprocessing and another step after composition. Before making the composition, if a probabilistic model contains compound formula nodes, we temporarily remove these nodes with all the edges coming in and out of them, and save them for later use. Now the probabilistic model just contains atomic nodes, and in each step of composition, we get an updated probabilistic model. Then we remove the restriction (which is only applicable in the single proof model case) that mergers only occur on the root nodes of a proof model and the atomic nodes of a probabilistic model, i.e., mergers can occur on both roots and compound nodes of proof models. (Since all the compound nodes in the original probabilistic model have been removed temporarily, any compound formula nodes found in the current probabilistic model in the process of composition should be from some previous integrated proofs.) In the part of the algorithm deciding whether a proof is eligible to be composed, we add one more check: if the proof involves the compound nodes already existing in the node set with temporarily removed nodes, we ignore this proof. (Otherwise it will bring duplicate nodes to the integrated model.) After the composition of all the eligible proofs, we add the removed compound nodes back to the probabilistic model. Because what we have changed in the original probabilistic model is just adding more children to its nodes, the probability distributions of all the nodes in the original model do not change.

5.3.2 Automation

In all the previous examples, both proof models and resulting composite models are very simple. They could be built even without the use of natural deductions and conversions. However, for a large domain problem, this might not be possible. For example, consider a knowledge base containing 1,000 logical rules.[8] Consider building a Bayesian network to meet new requirements in the task domain of the existing logical knowledge base. We can use the technique presented in this paper to automate the process of reusing the existing knowledge base. For example, we may still get the domain experts to specify the basic probabilistic model, but then use natural deduction proofs to augment this model automatically. Through the use of soft evidence updating, we can even assign uncertainty to logical rules.

Because of the mature and wide use of logical theories in the past decades, we have large logical knowledge bases to support logical reasoning in many task domains. One motivation of this paper is also to introduce a way of making use of existing logical knowledge bases to support modern approaches to probabilistic reasoning.

5.3.3 Probabilistic knowledge base

The goal of the approach introduced in this paper is not to find an equivalent Bayesian network to represent existing logical knowledge base. Because natural deductions are completely based on the logical knowledge base, when all the logical rules in the logical knowledge base are certain, the use of any generated proof model of a formula will produce the same result as direct logical reasoning. So for queries over formulas entailed by the logical knowledge

[8] The logical rules of such a large KB cannot be totally from human specification, and must be learned through some learning techniques.

base with certainty, translation to a proof model is redundant, as the queries can be answered by the logical system directly.

However, with the aid of composition, for queries over related formulas originally existing in the probabilistic model, this translation is very helpful. It makes possible for domain experts to specify different knowledge, based on its type: logical or probabilistic, in the most suitable way: logical formulas or Bayesian networks, while reasoning with a uniform model. The reasoning process is not based on the probabilistic model alone, but also complemented by the available logical knowledge. For some problems, this could be done by applying logical reasoning systems and probabilistic reasoning systems separately and sequentially. A simple example can be shown by Fig. 20, where a query regarding the posterior probability of C, denoted by $P(C|\mathbf{e})$, can be done by using the KB to reason about B at first and then applying the conclusion of B as hard evidence into the BN to get the posterior probability of C. Note that the KB contains three logical formulas: A, $A \rightarrow B$, and $C \rightarrow D$, and the BN just describes our knowledge about the probabilistic dependency between B and C. The result is same as directly using the composite model at the bottom of Fig. 20, with which, however, the user does not even have to build a real separate logic reasoning systems. Particularly, for a more complex probabilistic model, sequential uses of separate models could be cumbersome. For example, instead of only having B, if there are more nodes that could potentially affect the probability of C in the BN of Fig. 20, the user needs to figure out all of them at first, query the logical system secondly, and last return to BN to do probabilistic reasoning. With the translation and the composition, we automate the process and can take the advantage of separate specification of knowledge and uniform reasoning.

This approach makes special sense when we bring uncertainty into the logical knowledge base. There are two ways. One is based on assumptions. In this case, the pre-existing knowledge base has no uncertainty, but we add more assumption formulas into it. For example, these assumptions may come from the probabilistic model. Then they can be used in normal proof steps. This way has much higher flexibility for integrating logical and probabilistic reasoning and provides a much larger space of support for queries. For the same example of Fig. 20, if we are interested in $P(D|\mathbf{e})$, then the sequential use of two separate models cannot help, as the probabilistic output C of BN cannot be used to reason about D in the KB. If, in the KB, we add one formula C^* to represent the assumption of C, we obtain the composite model shown in Fig. 21, which we can use to reason about D's posterior probability.

The second way is based on making the logical knowledge uncertain. Uncertainty can occur on findings (or facts) or even logical rules, and can have different values at different times. In the case of Fig. 11, we are no longer sure

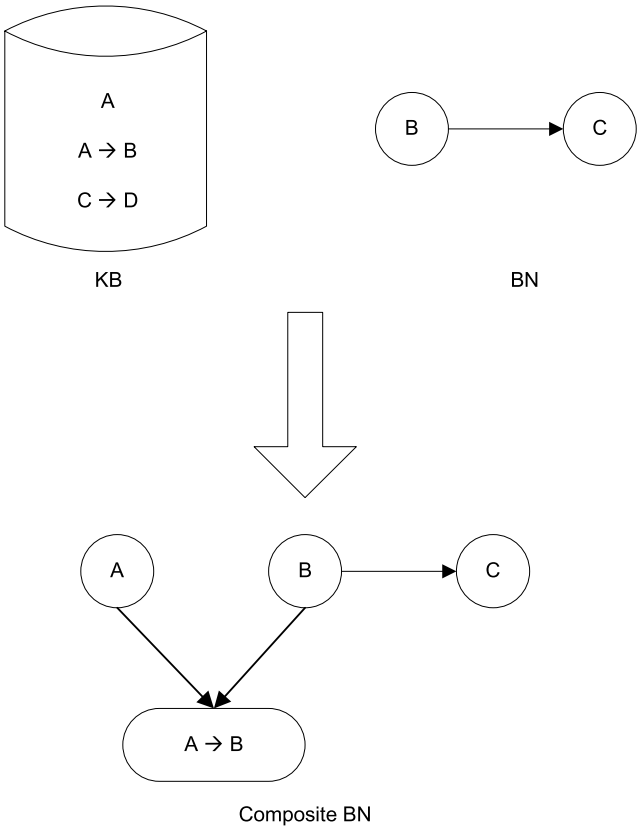

Fig. 20 The composite BN provides the same result as using separate models sequentially

that one of the rules holds. More generally, we can assign a probability to one or more facts or rules.

Of course these two ways of bringing uncertainty into the traditional logical knowledge base can be mixed. Therefore, given new evidence on some atomic formulas or even the logical rules, especially considering the use of soft evidence updating [52, 97], any influences from changes of uncertainty of these evidence nodes can be captured through the composite model, while traditional logical reasoning cannot handle uncertainty on formulas at all. In addition, such a way of translating provides high flexibility in building the model, as we can put any related formulas into the composite model for probabilistic reasoning regardless of their uncertainty (even if it is close to zero).

6 Related work

First-Order Logic (FOL), often referred as classical logic, is the formal system of logic that has been most extensively studied. Although it has great expressive power, FOL is still not sufficient and efficient to model all the ways humans reason, and therefore is also used as a foundation for defining extended logics.

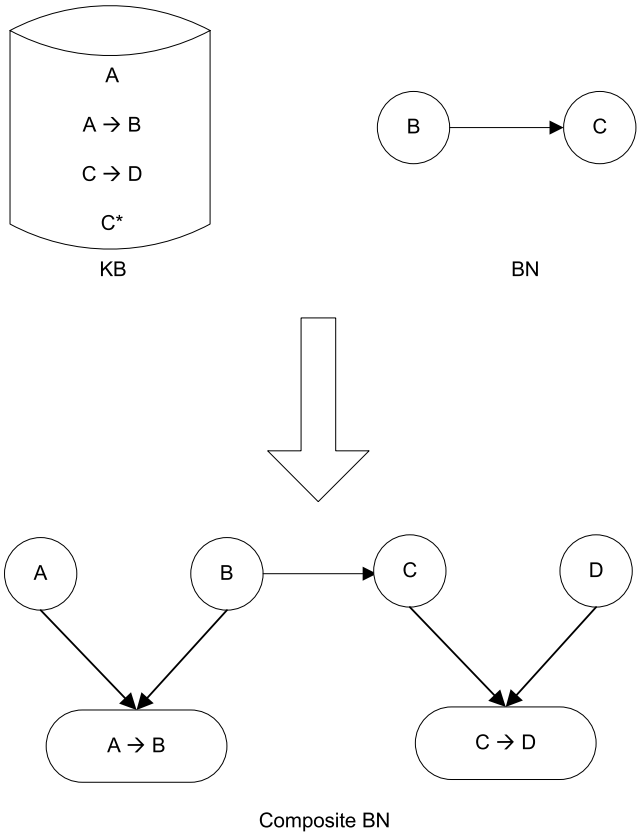

Fig. 21 The KB for the example of Fig. 20 with addition of assumption C, which is denoted by C^*

One important issue in classical logic is the lack of treatment of plausible reasoning under uncertainty. Uncertain conclusions result from fallible methods or uncertain premises and are natural in almost any real world application. Uncertainty reasoning is a realistic requirement for most modern expert systems, and has led to various research based on extension of classical logic.

6.1 First-order probabilistic models

One major extension of classical logic to reasoning under uncertainty is based on probability theory. Probabilistic models deal well with inherent uncertainty, while FOL has the power of rich expressiveness of knowledge. So the integration of these two kinds of inference is highly desirable. One of the earliest and most influential works in defining precise semantics for probabilistic logics is a paper by Nilsson [73] that computes lower and upper bounds instead of point values for the probabilities of queried sentences. Earlier work [26, 70] proposes a language for reasoning about probability and for interpreting truth values probabilistically. Bacchus and Halpern [3, 34] distinguish degrees of belief from relative frequencies and provide the semantics relating them. The popularity of Bayesian networks as a modeling tool starting from the mid-1980s attracted many researchers to study the integration of FOL with Bayesian networks for probabilistic reasoning. Recent research includes PRM [29, 31, 54], MEBN [55, 56], BLP [49], LBN [27], and RBN [43, 44]. Besides Bayesian approaches, other graphical models have also been proposed, such as MLN [83]. Logic program-based approaches include ICL [80] and SLP [67, 68]. We will discuss and compare this closely related work in more detail. For a better understanding of relations between the related approaches, we recommend two good surveys [66, 86] that analyze and categorize existing techniques in this field. In most of these approaches, probabilities, which are used as a measure of subjective belief, are assigned to propositions. This is the category our work belongs to, and in particular, we would put our approach in the branch of *unknown* at the bottom level of the taxonomy in [66], as we need to list all objects that might exist, although we have our special way of treating unknown objects as shown in Sect. 7.2.

6.1.1 *Approaches based on Bayesian networks*

Bayesian Logic Programs (BLPs) [49] are FOL-like programs that consist of a set of Bayesian definite clauses with quantitative information. A BLP specifies a Bayesian network, as each Bayesian predicate is more like a random variable, instead of logical predicate. A combination rule is used in BLPs to combine different ground instances of a rule with the same ground atom as head. As a variant of BLPs, Logical Bayesian Networks (LBNs) [27] explicitly distinguish logical knowledge and probabilistic knowledge through two disjoint sets of predicates: the set of logical predicates and the set of probabilistic predicates. The extra set of normal logical clauses specifies deterministic background information, which can be used to remove probability distribution that are not meaningful in the induced Bayesian network.

The way knowledge is represented in LBNs is very similar to our approach, as both have separate specifications for logical knowledge and probabilistic knowledge, and the logical knowledge in both approaches is used as the background knowledge. However, there are still important differences. In BLPs and LBNs, the ground logical atoms are excluded from the induced Bayesian network, as they are fixed to be deterministic and no uncertainty is considered over them unless the design is changed. In addition, random variables eligible to appear in the induced Bayesian network are only those ground Bayesian atoms in the least Herbrand model of the program. This means that queries about atoms outside the least Herbrand model will not be considered. In our approach, proof models could be built in different sizes, as described in the three scenarios introduced in the previous section, based on the realistic consideration of the degree of detail in knowledge representation and on computing capacity and efficiency, and then composed with the existing

probabilistic models, which are the center for extension. If the normal logical formulas turn out to be uncertain, we do not have to rebuild the composite model for probabilistic reasoning, but apply new evidence, which could be soft evidence. This way of using logical knowledge is more flexible than the way of using logical clauses in LBNs, because new findings or new uncertainties may be continuously added to a knowledge base; this is particularly useful in information analysis applications. We do not distinguish Bayesian atoms and logical atoms, as all the formulas are standard binary ones.[9] An atom included in the proof model could be an arbitrary formula in the knowledge base as long as it plays a role in some natural deduction proof. It is not even necessarily a goal that could be proved. In addition, compared with the definite clauses used in BLP and LBN, our approach accepts any standard FOL formulas.

A Multi-Entity Bayesian Network (MEBN) [55, 56] specifies a probabilistic knowledge base as a set of parameterized fragments of Bayesian networks (MFrag) [56], each of which describes probability information about a set of related random variables. This collection of Mfrags, called a MEBN Theory, defines a unique joint probability distribution over random variables in the theory if all the consistency constraints associated with each Mfrag are satisfied. Based on findings, Situation-Specific Bayesian Networks (SSBNs) are generated in response to specific queries. A special feature of MEBNs is the use of context constraints in each MFrag to restrict its instantiations. The constraints play a similar role to the logical literals of the conditional dependency clause in LBN. MEBN is a well defined and rich language that extends both FOL and Bayesian networks [55, 56]. Because of its generality, implementation of MEBN systems is difficult [86].

A Probabilistic Relational Model (PRM) [29, 31, 54] is a framework to represent probabilistic models over multiple entities and relations between them. Generally, a PRM specifies a relational schema for classes of objects and attributes, a structure to define the probabilistic dependencies between attributes, and the numerical probability information including aggregate functions. A database (skeleton) can be used to instantiate entities and relations, and construct the Bayesian network for reasoning. PRMs are based on the formalism of Frame Systems, which are less expressive than FOL.

In all the above approaches standard Bayesian networks are constructed from knowledge bases, a meta-approach called Knowledge Based Model Construction (KBMC). However, in our approach, the proof model extracted through natural deduction proofs, which represents the logical knowledge in a probabilistic form, is used as supplemental knowledge for the original probabilistic model, which pre-exists to represent our major interest and can actually be specified in any formalism, such as MEBN. This simplifies integration of logic knowledge with probabilistic reasoning, as both are in Bayesian network form. Regarding probability dependencies and distributions, for LBN, the edges of resulting Bayesian networks are determined from the *conditional dependency clauses* and the CPTs are from the *logical CPD*. PRMs have aggregate functions besides the ordinary CPDs. MEBNs use MFrags to specify each local probability dependencies and distributions. For complex relations, aggregation functions and combining rules must be used to combine influences. However, in our approach, the added edges and CPTs all depend on the logical structure of extracted logical formulas. The added nodes build the indirect connections between the ones of the original probabilistic model.

In addition, our approach explicitly distinguishes probabilistic knowledge and logical knowledge, and each kind of knowledge is represented through its most commonly used and familiar way: Bayesian networks and FOL formulas. However, the reasoning is still based on a uniform model. This way of integration enables separate knowledge specification in the design phase. Even FOL-conversant knowledge engineers who are not familiar with Bayesian approaches can use their previous knowledge to build a model that can later support probabilistic reasoning.

Further, most techniques are designed completely as a new formalism or variant of existing FOL or Bayesian networks to implement the integration of reasoning. In their real use, users have to study new way of designing and modeling. This conversion is hard and always comes with waste of efforts as many techniques are still in primary stage. Our approach is not a new language, but a framework, supported by a theorem prover, that can automatically transform the users' knowledge expressed in these two traditional formalisms into a uniform model for probabilistic reasoning. This makes our approach an alternative way of not only designing new expert systems but also making use of historical ones.

Also notice that our approach defines an extension of existing probabilistic model with logical rules. The resulting model used for reasoning is still a Bayesian network. If the original probabilistic dependencies do not exist, the extension effect is just to build a Bayesian network representing a logical knowledge base as all the entailments still hold. When the logical knowledge does not exist, there is no change to the original probabilistic model.

6.1.2 Non-Bayesian network based approaches

A Stochastic Logic Program (SLP) [68] consists of a set of labeled clauses, which are range-restricted definite clauses.

[9] We might have non-binary random variables in the composite model that come from the original probabilistic model. However, they will never get involved in proof and composition.

Probabilistic reasoning is done through a stochastic *SLD-tree*. The labels are probabilities. However, the probability distribution is not defined over proposition, but over proofs. In our approach, we have no restriction on the form of logic formulas. The probabilities are directly defined over propositions. We also use proofs, which are based on natural deduction. Proofs are not used for probabilistic reasoning, but for knowledge selection. We can compose the proof BN with the original Bayesian network, which may provide the prior probability distribution for root nodes. The final probabilistic reasoning is through a Bayesian network.

The Independent Choice Logic (ICL) [80] can be seen as adding independent stochastic inputs to a logic program, with a look similar to SLP. From another perspective, it can be regarded as a way of rule-based specification of Bayesian networks with logical variables. Compared with our approach, probabilities of ICL are specified only over alternatives, and probabilistic reasoning is done directly instead of through Bayesian network. The logic programs are not arbitrary. Atoms are classified to be atomic choice or not.

A Markov Logic Networks (MLN) [83] is a set of first-order formulas with weights attached. The MLN approach can be regarded as a special case of KBMC. Together with a finite set of constants, an MLN defines a ground Markov network, through which the reasoning is conducted. Unlike Bayesian networks, Markov networks are undirected networks. When complex constructs are present, inference in Markov networks can become quite slow [86]. MLNs have great expressive power and can be viewed as a generalization of FOL. However, learning and explaining the weights are still hard problems.

All the above techniques appear very close to normal logic programs, especially for SLP and MLN. These languages include direct support for arithmetic and probabilistic computation in their processors. However, the knowledge bases written in them are not pure FOL KBs, because they require the addition of numeric labels that are interpreted in a special way. Some are regarded as probabilities (in SLPs), while some are weights (in MLNs). Since natively FOL formulas are not designed for probabilistic reasoning, these labels are hard even for domain experts to specify, and also hard for users to understand and maintain. To overcome this difficult step, it has been suggested that learning be used as the source of the values of these labels, but learning itself is another hard problem, and a great deal of the latest work concentrates on it. For example, for MLN, recent papers address structure learning (formula learning) [6, 53], parameter learning (weight learning) [42, 61, 88] and both [41]. For PRM, learning algorithms for both structure and parameters have been proposed in [29, 31, 32]. For BLP, the learning principle is discussed in [48]. A search algorithm for learning LBN is proposed in [28], which is also valid for learning some other directed probabilistic logical models. One new method for parameter learning for relational models is introduced in [45], based on compiling a RBN model into a computational structure for the likelihood function and its partial derivatives. Some detailed discussions of current and emerging trends in learning can be found in [16, 30].

By contrast, in our approach, through natural deduction proofs the logical models are extracted directly from the logical knowledge base, which consists of standard FOL formulas that are not restricted to any special form. Additionally, no learning is involved in our framework, as we expect that the probabilistic knowledge is best presented through the probabilistic model Bayesian network.

Although our approach separates the logical and probabilistic knowledge specifications, we still allow the user to place uncertainty over logical rules, using the technique of soft evidence updating [52, 97], since each logical rule used in the proof has a corresponding node in our proof-based logical model, which is just a Bayesian network. This way of attaching uncertainty information over not only atomic formulas but the whole logic rules, is quite different from what is done in most other techniques, which generate probabilistic models that contain only atomic formulas as nodes, such as BLP/LBN [27, 49] and MLN [83].

As discussed before in Sect. 5.3.2, one motivation of our approach is to introduce a way of making use of existing logical knowledge bases to support modern approaches to probabilistic reasoning. This point of view, to the authors' knowledge, has never been considered in other work on the integration of logical and probabilistic reasoning.

From another perspective, compared with our approach of converting logical knowledge into Bayesian networks, there have also been some work done in the reverse direction, i.e., from Bayesian networks to logical theories [14]. However, the translation of a BN into a pure logical theory requires including axioms for arithmetic and for probability itself, which would take a lot of space, slow down computation, and be difficult for a user to read. For example, a set of axioms for a FOL theory of probability given in [3] includes the field axioms. A user would undoubtedly find the application of field axioms to probabilistic inference very hard to comprehend. So the resulting logical theory from the integration approach with translation in this direction should rather be regarded as supporting logic-based techniques to do probabilistic inference, instead of supporting human-oriented knowledge representation and reasoning. In fact, some authors have suggested using a translation of Bayesian networks to propositional logic as a step in probabilistic inference, which would be hidden from the user [14].

6.2 Models based on non-standard logics

Traditional tools based on probability theory still cannot handle all facets of uncertainty. This leads to extensions

of classical logic based on different representations of uncertainty, such as possibility theory and belief functions. The main motivation is a more faithful treatment of incomplete knowledge [21]. Among the vast literature, one important representative work is possibilistic logic [24, 25]. Like our approach, possibilistic logic is also based on two value states, i.e., Boolean propositions. However, the weights attached to sentences quantify their possibility or necessity. For a sentence p, the degrees of possibility and necessity are denoted by $\Pi(p)$ and $N(p)$ [18]. Dubois, Lang, and Prade [17, 18, 22] show how to extend the resolution principle to possibilistic logic. Ignorance is well expressed in possbilistic logic through $(N(p), N(\neg p)) = (0, 0)$, and the complete lack of certainty that p is false $(N(\neg p) = 0)$ can be clearly distinguished from the total certainty that p is true $(N(p) = 1)$. For consistent knowledge bases, we can conclude $N(\neg p) = 0$ given $N(p) = 1$, but not the other way around. This is quite different from the probability measures where $Prob(p) = 1$ is equivalent to $Prob(\neg p) = 0$ [24]. The fundamental semantic differences between probabilities and possibilities determine the different ways of handling uncertainty in probabilistic logic and possibilistic logic. For example, reasoning based on a possibilistic knowledge base uses only the part of knowledge one is most sure of [24]. "Possibilistic logic may especially be used for encoding user preferences, since possibility measures can actually be viewed as rankings (on worlds or also objects) along an ordinal scale" [93]. One preliminary investigation of the potentials of possibilistic logic in the representation and combination of preferences in decision analysis can be found in [4]. Possibilistic logic has been implemented in many ways, e.g., the POSLOG system [23]. The possibility measure was introduced by Zadeh [99] and was greatly developed by Dubois, Prade and others [19]. It has a representation whose complexity is linear with the number of elements in the set of possible worlds (the frame of discernment) [21]. It deals well with default reasoning and counterfactual reasoning [35].

The Dempster-Shafer (D-S) theory [87] (sometimes called belief function theory) provides an alternative way of handling partially specified models, unlike the pure Bayesian theory that requires the specification of a complete probabilistic model before reasoning can start. D-S theory computes probabilities of provability rather than computing probabilities of truth. The mass function m is defined for each possible proposition. m is also called a basic probability assignment, and the belief function *bel* is defined based on m. If we use W to represent the possible worlds and A to represent a proposition, the following expressions show the relations between m and *bel*:

$$\sum_{A \subseteq W} m(A) = 1,$$

$$bel(A) = \sum_{B \subseteq A} m(B).$$

In addition, *Dempster's Rule of Combination* provides a way to combine the impact of several pieces of evidence that are independent. For example, for two mass functions m_1 and m_2, we have:

$$m_1 \oplus m_2(A) = \frac{1}{c} \sum_{A_1 \cap A_2 = A} m(A_1) m(A_2),$$

where c is a normalization factor. This rule of combination is commutative and associative. Although these features make D-S theory attractive, problems do exist in its real use. One major one is computational complexity, which increases exponentially with respect to the size of the frame of discernment [75]. There are also other methods of computing belief functions, such as the Incidence Calculus [7, 58, 59], which is based on a representation of uncertainty using sets of points. Probabilities are not directly associated with formulas, but with possible worlds.

Vagueness and fuzziness lead to the extension of classical logic to many-valued logics and fuzzy logics. For example, in three-valued logic, besides the usual truth value true and false, the third value is to express the idea of "unknown" or "possible." Note that vague/fuzzy propositions are truth-functional, while the degree of uncertainty is not with respect to all the connectives [20, 24]. Many-valued logics normally consider degree of truth, instead of degree of uncertainty. One widely studied class of many-valued logics is annotated logic, which is proposed by Subrahmanian in [96] as a formal model to represent and reason under incomplete and inconsistent information, and it therefore addresses important aspects of the management of uncertainty in information analysis, although not the issue of representation and management of belief in hypotheses, which is the focus of our paper. Annotated logic has been greatly extended to support temporal and uncertain reasoning [95]. Related research can be found in, e.g., [1, 2, 9, 15, 50, 51, 63, 96]. Many-valued logics have been extended and applied in many fields, e.g., Kleene's three-valued logic is used in [74] for representing and reasoning with temporal information, and a new symbolic approach to representation of imprecise (or fuzzy) information has been proposed in [10] based on a symbolic many-valued logic. A more complete list of categorized references for the management of imperfect information can be found in [62, 93].

The comparison between our approach and these techniques is mainly a comparison of probability theory with possibility theory, belief function theory, and many-valued

logic. However, since they are just different representations of uncertainty, it is hard to conclude that approaches based on one representation are absolutely better for reasoning than ones based on the other. The choice should depend on the application domain. Probability has the advantage of being better studied and understood, and many arguments suggest that it is the only "rational" way to represent uncertainty under certain assumptions [35]. In particular, for applications in information analysis, such as hypothesis management, surprise detection, and generation of alerts, it is convenient to leverage probabilities to compute expected utilities and base decisions on the maximum expected utility principle. More detailed comparisons and discussions about these theories are available, e.g., in [35, 89].

Note that it may not be appropriate to simply classify uncertainty reasoning into probabilistic approaches or not. Some non-classical logics are better suited to support uncertain reasoning directly in particular domains. Some examples include logics for temporal and spatial reasoning. As another example, Description Logics (DLs) play an important role in the semantic web domain, and incorporating uncertainty in DL reasoning has been the topic of much research. DLs could be fuzzy [90–92, 94], probabilistic [64, 65], and possibilistic [39, 81]. A recent proposal uses a single reasoning procedure for dealing with uncertainty represented by different mathematical formalisms [33].

7 Correctness proofs

7.1 Correctness of the logical model

We want to demonstrate that models constructed as described above behave according to our logical intuitions. We consider an arbitrary "mixed" model of the sort in Fig. 10.

Note that the *roots* (nodes without parents) of the proof Bayesian network need not be atomic formulas, and not all subformulas of a logical node formula need to be present in the proof Bayesian network. For example, we might have proved $R \wedge Q$ from $(R \wedge Q) \wedge S$ without introducing the atomic formulas R and Q. In this case, S does not appear in the proof Bayesian network. Although it is not necessary to do so, it is convenient for this presentation to extend the proof Bayesian network so that all complex formulas that are not roots have exactly two parents. Also, we define $\neg A$ to be an abbreviation for $A \rightarrow \bot$.

We permit the causal Bayesian network to introduce arbitrary dependencies between the root nodes of the logical Bayesian network. We make the restriction that nodes of the causal Bayesian network that are not shared with the proof Bayesian network may not be part of the language of the proof Bayesian network. The proof Bayesian network just described, for example, could not be joined with a causal Bayesian network that refers to R, because R is in the proof Bayesian network but would not be properly linked. This can always be overcome by extending the proof Bayesian network so that it does contain R before joining with the causal Bayesian network.

As described in Sect. 4.2.1, the general claim that we wish to prove is that for an arbitrary probability distribution P over the language of the proof and for an arbitrary formula G occurring in the proof dependent on assumptions $A_1, \ldots, A_n$, $P(G = \mathsf{true} \mid A_1 = \mathsf{true}, \ldots, A_n = \mathsf{true}) = 1$. For simplicity, we will use $P(a_i)$ to abbreviate $P(A_i = \mathsf{true})$ and $P(\bar{a_i})$ to abbreviate $P(A_i = \mathsf{false})$. Let $\mathcal{L}$ represent the set of all nodes of the causal Bayesian network and all roots of the proof Bayesian network. We let P be an arbitrary probability distribution over $\mathcal{L}$ satisfying the independence relations induced by the DAG, with the restriction that $P(\bot) = 0$. Our first step will be to extend P to all logical formulas over $\mathcal{L}$.

Let $\mathbf{v}$ be an arbitrary function from $\mathcal{L}$ to $\{\mathsf{true}, \mathsf{false}\}$. For an arbitrary complex formula A, we define $\mathbf{v}[A]$ recursively on the structure of A by case depending on the main connective of A. If $A \equiv B \wedge C$ then we define $\mathbf{v}[A] = \mathsf{true}$ if $\mathbf{v}[B] = \mathsf{true}$ and $\mathbf{v}[C] = \mathsf{true}$; $\mathbf{v}[A] = \mathsf{false}$ otherwise. Similarly, $\mathbf{v}[B \vee C] = \mathsf{true}$ iff either of $\mathbf{v}[B]$ or $\mathbf{v}[C]$ are true, $\mathbf{v}[B \rightarrow C] = \mathsf{true}$ iff $\mathbf{v}[B] = \mathsf{false}$ or $\mathbf{v}[C] = \mathsf{true}$, and $\mathbf{v}[\bot] = \mathsf{false}$. When $\mathbf{v}[A] = \mathsf{true}$ we write $\mathbf{v} \models A$ and otherwise we write $\mathbf{v} \not\models A$. $\mathbf{v}$ is called a *valuation* over $\mathcal{L}$.

Using the set of all valuations over $\mathcal{L}$ as a sample space with all possible subsets as events, we can define a probability distribution P' such that, for any $A \in \mathcal{L}$, $P'(\{\mathbf{v} \mid \mathbf{v} \models A\}) = P(a)$. Since P is given for all $A \in \mathcal{L}$, P uniquely determines P' over any subset of valuations. We now use P to denote P', since this usage is unambiguous. Note in particular that $P(\bot) = P(\{\mathbf{v} \mid \mathbf{v} \models \bot\}) = P(\emptyset) = 0$ as required.

Since the composite DAG is, in fact, a Bayesian network, and since P is given over the parents of all logical nodes, the full distribution $\hat{P}$ over the entire DAG is uniquely determined. Since $\hat{P}$ extends P, we again use P to denote it. We next show that for every node A in the Bayesian network, $P(a) = P(\{\mathbf{v} \mid \mathbf{v} \models A\})$, by induction over the complexity of A. Complexity here means the depth of the parse tree of the formula relative to $\mathcal{L}$ (for example, the complexity of $B \wedge C$ is one greater than the maximum complexity of B or C), where all formulas in $\mathcal{L}$ are given complexity 1 regardless of their structure.

We have established the claim already for $A \in \mathcal{L}$, satisfying the base case, in which the complexity of A is 1. For the inductive step we suppose that A is of higher complexity. Suppose $A \equiv B \wedge C$. A must have parents in the proof Bayesian network in order that $A \notin \mathcal{L}$, so B and C are both nodes in the proof Bayesian network. By the inductive hypothesis, $P(b) = P(\{\mathbf{v} \mid \mathbf{v} \models B\})$ and $P(c) = P(\{\mathbf{v} \mid \mathbf{v} \models$

$C\}$). By the conditional probability table for $\wedge$,

$$\begin{aligned} P(B \wedge C = \mathsf{true}) &= P(b, c) \\ &= P(\{\mathbf{v} \mid \mathbf{v} \models B\}, \{\mathbf{v} \mid \mathbf{v} \models C\}) \\ &= P(\{\mathbf{v} \mid \mathbf{v} \models B\} \cap \{\mathbf{v} \mid \mathbf{v} \models C\}) \\ &= P(\{\mathbf{v} \mid \mathbf{v} \models B \text{ and } \mathbf{v} \models C\}) \\ &= P(\{\mathbf{v} \mid \mathbf{v} \models B \wedge C\}) \end{aligned}$$

If $A \equiv B \vee C$, then by the conditional probability table for $\vee$,

$$\begin{aligned} P(B \vee C = \mathsf{true}) &= P(b, c) + P(b, \bar{c}) + P(\bar{b}, c) \\ &= P(\{\mathbf{v} \mid \mathbf{v} \models B \text{ and } \mathbf{v} \models C\}) \\ &\quad + P(\{\mathbf{v} \mid \mathbf{v} \models B \text{ and } \mathbf{v} \not\models C\}) \\ &\quad + P(\{\mathbf{v} \mid \mathbf{v} \not\models B \text{ and } \mathbf{v} \models C\}) \\ &= P(\{\mathbf{v} \mid \mathbf{v} \models B \cup \mathbf{v} \models C\}) \\ &= P(\{\mathbf{v} \mid \mathbf{v} \models B \vee C\}) \end{aligned}$$

The case in which $A \equiv B \rightarrow C$ is similar.

We have thus established that for every node A in the composite network, $P(a) = P(\{\mathbf{v} \mid \mathbf{v} \models A\})$, indicating that the semantics of our network is what we would expect. Next consider any formula occurrence G in the proof and let $\Gamma = \{A_1, \ldots, A_n\}$ be the set of all assumptions on which G depends. Our original goal was to show that $P(g \mid \Gamma) = 1$. Since $\Gamma \vdash G$, the soundness of the logical rules gives us that for every $\mathbf{v}$ such that $\mathbf{v} \models \Gamma$, $\mathbf{v} \models G$. Thus $\{\mathbf{v} \mid \mathbf{v} \models \Gamma, G\} = \{\mathbf{v} \mid \mathbf{v} \models \Gamma\}$. This allows us to reason:

$$P(g \mid \Gamma) = \frac{P(g, \Gamma)}{P(\Gamma)} = \frac{P(\{\mathbf{v} \mid \mathbf{v} \models \Gamma, G\})}{P(\{\mathbf{v} \mid \mathbf{v} \models \Gamma\})} = 1$$

The correctness proof demonstrates the desirable characteristic that the conclusion of a proof is in state "true" with probability 1, conditional on the premises of the proof being in the state "true." More generally, it demonstrates that any formula in the proof is true with probability 1, conditional on those premises on which it depends in the proof. One possible use of this is to observe that a given context node of the network is true; then every formula of the proof within that context will have value true as well.

Although we leave the "background context" Γ implicit when translating the proof in Fig. 6, we could have introduced a node for Γ as well, with all premises of the proof pointing to that node. Making a hard evidential observation on that node (which is only true when *all* of its parents are) then forces the truth of all conclusions that occur within that context, just as in the other contexts. Making a soft evidential observation on that node can provide a way to quantify that we have a certain degree of trust for the information that comes from a particular knowledge base. If we are reasoning over knowledge from many sources, we can assign a different context node for each source and quantify the trustworthiness of each source in this way. Note that this cannot be accomplished by assigning the appropriate degree of belief to each premise individually, because doing this indicates that the premises succeed or fail independently rather than together. A conclusion that depends on a very large number of highly likely premises will mistakenly be given a lower probability than it would be given when the context is used to indicate the single degree of certainty for the set of premises.

Another property of the networks that is made apparent by the correctness proof is the desirable semantic interpretation that the probability for a given node is the total probability of the set of models of that node. This is similar to the technique presented in [76] and allows one to define arbitrary distributions over logical models, while maintaining consistency with the Bayesian networks derived from proofs.

In some cases it might be desirable to force all atomic formulas to be explicitly represented in proofs. This can be done straightforwardly by converting the natural deduction proof to its so-called *long $\beta\eta$-normal form*. Certain search strategies for natural deduction proving will automatically generate such proofs without the need for conversion [8].

7.2 The first-order logic case

Although all the examples shown in the previous sections are propositional, our approach work in FOL case too. The proof for the first-order case is similar to the proof for the propositional case. We use a standard term models as used to prove completeness for first order logic. For any term t, we call $A(t)$ a subformula of $(\forall x)A(x)$ and of $(\exists x)A(x)$. The parents of any quantified node include all of its subformulas which occur in the proof and also a special syntactic construction of the form $A(x)_{s,\ldots,t}$ which is intended to represent all possible instantiations of x other than $s, \ldots, t$. This single node allows to avoid attempting to enumerate or sample from all terms in the language. The details of this proof are not presented in this paper.

7.3 Correctness of adding a new proof to the updated probabilistic model

We only provide a proof sketch for the propositional case; the FOL case is very similar. Starting from a probabilistic model Bayesian network B, we denote the first integrated proof as $G1$ and the second (current) proof as $G2$. B contains all the roots in $G1$ and $G2$ (we extend B if necessary). Then, there are two possibilities:

1. $G1$ and $G2$ do not have common compound formula nodes, as shown in Fig. 22.

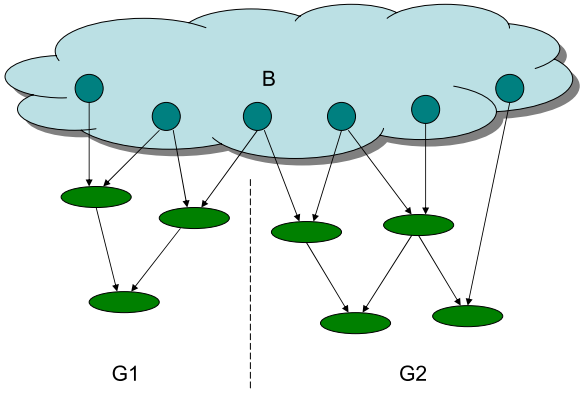

Fig. 22 The case when two proofs do not have common compound formula nodes

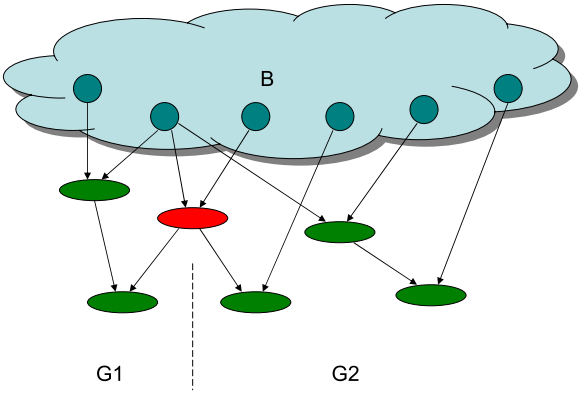

Fig. 23 The case when two proofs have common compound formula nodes

2. $G1$ and $G2$ have common compound formula nodes, as shown in Fig. 23. (The special case in which all the compound formula nodes are common is the same as adding one proof only, and the correctness is obvious.)

We extend $G1$ to be $BN(G1, G2)$, which denotes the Bayesian network having all the nodes in $G1$ and $G2$, including both root atomic formula nodes and compound formula nodes. We can see that all the root nodes of $BN(G1, G2)$ are also in B. $BN(G1, G2)$ is still a proof model that contains a complete set of nodes used in a proof for some given goal formula. In particular, here the number of goal formulas could be one or two, depending on we are composing two proofs of one same goal or two different goals, and each node only appears once even if it is used in two different proofs (no duplicate nodes). Also we denote the composite model as $BN(B, BN(G1, G2))$.

We then can define the same $\mathcal{L}$ and P as in Sect. 7.1: $\mathcal{L}$ represents the set of all the nodes of B and P is an arbitrary probability distribution over $\mathcal{L}$ satisfying the independence relations induced by B. In a way similar to Sect. 7.1, we can easily extend P to all the nodes of $BN(B, BN(G1, G2))$ and prove that for an arbitrary formula G occurring in the proof dependent on assumptions $A_1, \ldots, A_n$, $P(G = \text{true} \mid A_1 = \text{true}, \ldots, A_n = \text{true}) = 1$.

If we keep on adding different proofs to the current probabilistic model, following the same analysis as above, the resulting Bayesian network always keeps the same characteristic, i.e. the conclusion of a proof is in state "true" with probability 1, conditional on the premises of the proof being in the state "true."

8 Implementation considerations

We decided to use the IKL Lisp-like language to represent formulas within a proof [36–38]. An issue that had to be resolved is that of representing the proof in a convenient machine-readable form. Due to the prevalence of XML, we decided to use a variation of the XML format used in the Vampire theorem prover [82]. Since Vampire is a resolution theorem prover, while we use natural deduction, we modified the schema by allowing for an explicit representation of the rule used and of the *context*, defined as the set of assumptions, used in a proof step. The XML document representing the brown liquids proof contains a list of contexts

```
<contexts>
        <context id='1'>
                <formula>
                        (C)
                </formula>
        </context>

        <context id='2'>
                <formula>
                        (T)
                </formula>
        </context>
</contexts>

<proofSteps>
        <proofStep id='1'>
                <rule>
                        if E
                </rule>

                <premises>
                        <formula contextId='1'>
                                (C)
                        </formula>
                        <formula>
                                (if (C) (B))
                        </formula>
                </premises>
                <conclusion>
                        <formula>
                                (B)
                        </formula>
                </conclusion>
        </proofStep>
...
```

Fig. 24 A natural deduction proof in XML format

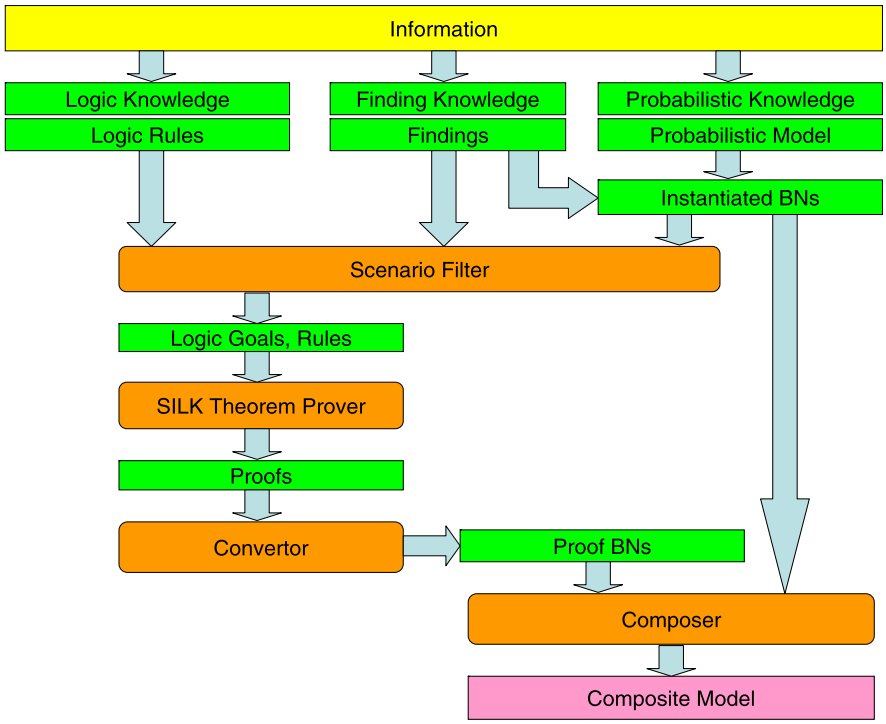

Fig. 25 The BALER software process flow [40]

and the proof steps, which are both numbered and refer to the contexts, as shown in Fig. 24. Note that the IKL syntax requires what seems to be, in the propositional case, superfluous parentheses.

The methodology described in this paper has been partly implemented in the BALER software system, which is described in Fig. 25. In this paper, we do not discuss the use of OWL for the representation of variables (nodes in Bayesian networks, their attributes, and variables of the logical theories) that are used to support the composition process. See [40] for some discussions of this topic. A full implementation and an empirical study of the ergonomical aspects of our methodology have not yet carried out and should be the subject of future work.

9 Conclusion

In this paper, we describe a framework for systems that incorporate the advantages of both Bayesian and logical systems. We define a formalism for the conversion of automatically generated natural deduction proof trees into Bayesian networks. We then demonstrate that the merging of such networks with domain-specific causal models forms a consistent Bayesian network with correct values for the formulas derived in the proof. We have used examples to show the influence brought by logical knowledge to uncertainty reasoning in our integrated approach and have discussed its extensions that address practical issues.

Depending on problems, we can build composite models of different sizes automatically by adding fewer or more proofs as needed. Especially for large pre-existing knowledge bases, such automation facilitates building models of information analysis and, through the use of pre-existing logical knowledge, makes our specification more complete and objective. Because we are using Bayesian networks for probabilistic reasoning, the actual inference would be done only on the subgraph consisting of the query and evidence nodes and their ancestors [71], even if the model is very large. In particular, the complementary knowledge is always added as child nodes in resulting Bayesian networks. When there is no logical knowledge available as hard or soft evidence, the composite model just resumes to be the originally manual-specified Bayesian network. Meanwhile, our approach is very easy to understand. Based on support from a theorem prover, the whole process including conversion and composition is straightforward. In addition, our approach can be combined with any other Bayesian network-based techniques used for probabilistic reasoning. For example, we can use the Bayesian networks that are the output of other techniques as the input probabilistic models for our technique, instead of specifying them manually.

Acknowledgements This work was funded in part by the Disruptive Technology Office (DTO) Collaboration and Analyst System Effectiveness (CASE) Program, contract FA8750-06-C-0194 issued by Air Force Research Laboratory (AFRL). The views and conclusions are those of the authors, not of the US Government or its agencies. The authors thank the anonymous reviewers for many useful suggestions.

References

1. Abe J, Akama S (2001) On some aspects of decidability of annotated systems. In: Proceedings of the international conference on artificial intelligence, pp 789–795
2. Akama S, Abe J (1998) Many-valued and annotated modal logics. In: Proceedings of the 28th international symposium on multiple-valued logic. IEEE Computer Society, Washington, pp 114–119
3. Bacchus F (1990) Representing and reasoning with probabilistic knowledge: a logical approach to probabilities. MIT Press, Cambridge
4. Benferhat S, Dubois D, Prade H (2001) Towards a possibilistic logic handling of preferences. Appl Intell 14:303–317
5. Bertelè U, Brioschi F (1972) Nonserial dynamic programming. Academic Press, New York
6. Biba M, Ferilli S, Esposito F (2008) Discriminative structure learning of Markov logic networks. In: Proceedings of the 18th international conference on inductive logic programming, ILP '08. Springer, Berlin, pp 59–76
7. Bundy A (1985) Incidence calculus: a mechanism for probabilistic reasoning. J Autom Reason 1(3):263–283
8. Byrnes J (1999) Proof search and normal forms in natural deduction. PhD thesis, Department of Philosophy, Carnegie Mellon University
9. Carbogim DV, da Silva FSC (1998) Annotated logic applications for imperfect information. Appl Intell 9:163–172
10. Chachoua M, Pacholczyk D (2000) A symbolic approach to uncertainty management. Appl Intell 13:265–283
11. Cheng J, Emami R, Kerschberg L, Santos JE, Zhao Q, Nguyen H, Wang H, Huhns M, Valtorta M, Dang J, Goradia H, Huang J, Xi S (2005) Omniseer: a cognitive framework for user modeling, reuse of prior and tacit knowledge, and collaborative knowledge services. In: Proceedings of the 38th Hawaii international conference on system sciences (HICSS38), Big Island, HI
12. Cooper GF (1987) Probabilistic inference using belief networks is np-hard, memo KSL-87-27 (revised July 1988). Tech rep, Medical Computer Science Group, Knowledge Systems Laboratory, Stanford University
13. Cooper GF (1990) The computational complexity of probabilistic inference using Bayesian belief networks. Artif Intell 42:393–405
14. Darwiche A (2009) Modeling and reasoning with Bayesian networks. Cambridge University Press, Cambridge
15. Dekhtyar A, Subrahmanian VS (2000) Hybrid probabilistic programs. J Log Program 43(3):187–250
16. Dietterich TG, Domingos P, Getoor L, Muggleton S, Tadepalli P (2008) Structured machine learning: the next ten years. Mach Learn 73:3–23
17. Dubois D, Prade H (1987) Necessity measures and the resolution principle. IEEE Trans Syst Man Cybern 17:474–478
18. Dubois D, Prade H (1988) Possibility theory. Plenum Press, New York
19. Dubois D, Prade H (1990) An introduction to possibilistic and fuzzy logics. In: Readings in uncertain reasoning. Morgan Kaufmann Publishers, San Francisco, pp 742–761
20. Dubois D, Prade H (1994) Can we enforce full compositionality in uncertainty calculi? In: Proc of the 11th nat conf on artificial intelligence (AAAI-94). AAAI Press/MIT Press, Menlo Park/Cambridge, pp 149–154
21. Dubois D, Prade H (2001) Possibility theory, probability theory and multiple-valued logics: a clarification. Ann Math Artif Intell 32(1–4):35–66
22. Dubois D, Lang J, Prade H (1987) Theorem proving under uncertainty: a possibility theory-based approach. In: IJCAI'87: proceedings of the 10th international joint conference on artificial intelligence. Morgan Kaufmann Publishers Inc, San Francisco, pp 984–986
23. Dubois D, Lang J, Prade H (1990) Poslog, an inference system based on possibilistic logic. In: Proc of the North American fuzzy information processing society conference (NAFIPS'90): quarter century of fuzzyness, Toronto, Canada, 06/06/90–08/06/90, pp 177–180
24. Dubois D, Lang J, Prade H (1994) Automated reasoning using possibilistic logic: semantics, belief revision and variable certainty weights. IEEE Trans Knowl Data Eng 6(1):64–71
25. Dubois D, Lang J, Prade H (1994) Possibilistic logic. In: Gabbay DM, Hogger CJ, JA Robinson (eds) Handbook of logic in artificial intelligence and logic programming. Nonmonotonic reasoning and uncertain reasoning, vol 3. Oxford University Press, New York, pp 439–513
26. Fagin R, Halpern JY, Megiddo N (1990) A logic for reasoning about probabilities. Inf Comput 87:78–128
27. Fierens D, Blockeel H, Bruynooghe M, Ramon J (2005) Logical Bayesian networks and their relation to other probabilistic logical models. In: Proceedings of the 15th international conference on inductive logic programming. Springer, Berlin, pp 121–135
28. Fierens D, Ramon J, Bruynooghe M, Blockeel H (2008) Learning directed probabilistic logical models: ordering-search versus structure-search. Ann Math Artif Intell 54:99–133
29. Friedman N, Getoor L, Koller D, Pfeffer A (1999) Learning probabilistic relational models. In: IJCAI. Springer, Berlin, pp 1300–1309
30. Getoor L, Taskar B (2007) Introduction to statistical relational learning (adaptive computation and machine learning). MIT Press, Cambridge
31. Getoor L, Friedman N, Koller D, Taskar B (2001) Learning probabilistic models of relational structure. In: Proceedings of the eighteenth international conference on machine learning. Morgan Kaufmann, San Mateo, pp 170–177
32. Getoor L, Friedman N, Koller D, Taskar B (2002) Learning probabilistic models of link structure. J Mach Learn Res 3:679–707
33. Haarslev Pai HI V, Shiri N (2009) A formal framework for description logics with uncertainty. Int J Approx Reason 50(9):1399–1415
34. Halpern JY (1990) An analysis of first-order logics of probability. Artif Intell 46:311–350
35. Halpern JY (2003) Reasoning about uncertainty. MIT Press, Cambridge
36. Delugach H (ed) (2005) Common logic—a framework for a family of logic-based languages. Tech rep, International Standards Organization: iSO/IEC JTC 1/SC 32N1377, International Standards Organization Final Committee Draft, 2005-12-13, available at http://i-cl.tamu.edu/docs/cl/32N1377T-FCD24707.pdf
37. Hayes PJ (2006) IKL guide. Tech rep, Florida Institute for Human and Machine Cognition, unpublished memorandum available at http://www.ihmc.us/users/phayes/IKL/GUIDE/GUIDE.html
38. Hayes PJ, Menzel C (2006) IKL specification document. Tech rep, Florida Institute for Human and Machine Cognition, unpublished memorandum available at http://www.ihmc.us/users/phayes/IKL/SPEC/SPEC.html
39. Hollunder B (1995) An alternative proof method for possibilistic logic and its application to terminological logics. Int J Approx Reason 12(2):85–109
40. Huhns M, Valtorta M, Wang J (2010) Design principles for ontological support of Bayesian evidence management. In: Obrst L, Janssen T, Ceusters W (eds) Semantic technologies, ontologies, and information sharing for intelligence analysis. IOS Press, Amsterdam, pp 163–178
41. Huynh TN, Mooney RJ (2008) Discriminative structure and parameter learning for Markov logic networks. In: Proceedings of the 25th international conference on machine learning, ICML '08. ACM, New York, pp 416–423

42. Huynh TN, Mooney RJ (2009) Max-margin weight learning for Markov logic networks. In: Proceedings of the European conference on machine learning and principles and practice of knowledge discovery in databases (ECML/PKDD-09). Bled, pp 248–263
43. Jaeger M (1997) Relational Bayesian networks. In: Proceedings of the 13th conference of uncertainty in artificial intelligence (UAI-13). Morgan Kaufmann, San Mateo, pp 266–273
44. Jaeger M (2002) Relational Bayesian networks: a survey. Electron Trans Artif Intell 6
45. Jaeger M (2007) Parameter learning for relational Bayesian networks. In: Proceedings of the international conference in machine learning
46. Jensen FV, Nielsen TD (2007) Bayesian networks and decision graphs, 2nd edn. Springer, New York
47. Johnson DS (1990) A catalog of complexity classes. In: van Leeuwen J (ed) Handbook of theoretical computer science, vol. A: algorithms and complexity. MIT Press, Cambridge, pp 67–161
48. Kersting K, De Raedt L (2008) Basic principles of learning Bayesian logic programs. In: De Raedt L, Frasconi P, Kersting K, Muggleton S (eds) Probabilistic inductive logic programming. Springer, Berlin, pp 189–221
49. Kersting K, Raedt LD (2001) Bayesian logic programs. CoRR cs.AI/0111058
50. Kifer M, Subrahmanian VS (1989) On the expressive power of annotated logic programs. In: Proceedings of the North American conference on logic programming, pp 1069–1089
51. Kifer M, Subrahmanian VS (1992) Theory of generalized annotated logic programming and its applications. J Log Program 12:335–367
52. Kim YG, Valtorta M, Vomlel J (2004) A prototypical system for soft evidential update. Appl Intell 21(1):81–97
53. Kok S, Domingos P (2009) Learning Markov logic network structure via hypergraph lifting. In: Proceedings of the 26th international conference on machine learning (ICML-09)
54. Koller D (1998) Pfeffer a probabilistic frame-based systems. In: Proc AAAI. AAAI Press, Menlo Park, pp 580–587
55. Laskey KB (2006) First-order Bayesian logic. Technical report C4I06-01. Tech rep, SEOR Department, George Mason University
56. Laskey KB (2008) MEBN: a language for first-order knowledge bases. Artif Intell 172:140–178
57. Laskey KB, Mahoney SM (1997) Network fragments: representing knowledge for constructing probabilistic models. In: Proceedings of the thirteenth annual conference on uncertainty in artificial intelligence (UAI-97), Providence, pp 334–341
58. Liu W, Bundy A (1994) A comprehensive comparison between generalized incidence calculus and the Dempster-Shafer theory of evidence. Int J Hum-Comput Stud 40:1009–1032
59. Liu W, McBryan D, Bundy A (1998) The method of assigning incidences. Appl Intell 9:139–161
60. Loveland DW, Stickel M (1976) A hole in goal trees: Some guidance from resolution theory. IEEE Trans Comput 25:335–341
61. Lowd D, Domingos P (2007) Efficient weight learning for Markov logic networks. In: Proceedings of the eleventh European conference on principles and practice of knowledge discovery in databases, pp 200–211
62. Loyer Y, Straccia U (2009) Approximate well-founded semantics, query answering and generalized normal logic programs over lattices. Ann Math Artif Intell 55:389–417
63. Lu JJ, Murray NV, Rosenthal E (1993) Signed formulas and annotated logics. In: Proceedings of int symposium on multiple-valued logic, pp 48–53
64. Lukasiewicz T (2007) Probabilistic description logic programs. Int J Approx Reason 45(2):288–307
65. Lukasiewicz T (2008) Probabilistic description logic programs under inheritance with overriding for the semantic web. Int J Approx Reason 49(1):18–34
66. Milch B, Russell S (2007) First-order probabilistic languages: Into the unknown. In: Proceedings of the 16th international conference on inductive logic programming, pp 10–24
67. Muggleton S (1996) Stochastic logic programs. In: Advances in inductive logic programming. IOS Press, Amsterdam, pp 254–264
68. Muggleton S (2000) Learning stochastic logic programs. In: Getoor L, Jensen D (eds) Proceedings of the AAAI2000 workshop on learning statistical models from relational data, URL: http://www.doc.ic.ac.uk/~shm/Papers/slplearn.pdf
69. Neapolitan RE (1990) Probabilistic reasoning in expert systems: theory and algorithms. Wiley, New York
70. Ng R, Subrahmanian VS (1992) Probabilistic logic programming. Inf Comput 101:150–201
71. Ngo L, Haddawy P (1996) Answering queries from context-sensitive probabilistic knowledge bases. Theor Comput Sci 171:147–177
72. Niles I, Pease A (2001) Towards a standard upper ontology. In: Welty C, Smith B (eds) Proceedings of the 2nd international conference on formal ontology in information systems (FOIS-2001), Ogunquit, ME, USA, pp 2–9
73. Nilsson NJ (1986) Probabilistic logic. Artif Intell 28(1):71–87
74. Obeid N (2005) A formalism for representing and reasoning with temporal information, event and change. Appl Intell 23:109–119
75. Orponen P (1990) Dempster's rule of combination is #p-complete. Artif Intell 44:245–253
76. Paris J (1994) The uncertain reasoner's companion: a mathematical perspective. Cambridge tracts in theoretical computer science, vol 39. Cambridge University Press, Cambridge
77. Park J (2002) Map complexity results and approximation methods. In: Proceedings of the 18th annual conference on uncertainty in artificial intelligence (UAI-02). Morgan Kaufmann, San Francisco, pp 388–439
78. Pearl J (2000) Causality: modeling, reasoning, and inference. Cambridge University Press, Cambridge
79. Peng Y, Reggia JA (1990) Abductive inference models for diagnostic problem solving. Springer, New York
80. Poole D (2008) The independent choice logic and beyond. In: Probabilistic inductive logic programming: theory and applications. Springer, Berlin, pp 222–243
81. Qi G, Pan JZ, Ji Q (2007) A possibilistic extension of description logics. In: Proceedings of the international workshop on description logics (DL'07), pp 435–442
82. Riazanov A, Voronkov A (2002) The design and implementation of Vampire. AI Commun. 15:91–110
83. Richardson M, Domingos P (2006) Markov logic networks. Mach Learn 62(1–2):107–136
84. Rose DJ (1972) A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In: Read R (ed) Graph theory and computing. Academic Press, New York, pp 183–217
85. Roth D (1996) On the hardness of approximate reasoning. Artif Intell 82:273–302
86. de Salvo Braz R, Amir E, Roth D (2008) A survey of first-order probabilistic models. In: Holmes D, Jain L (eds) Innovations in Bayesian networks. Springer, Berlin, pp 289–317. URL: http://l2r.cs.uiuc.edu/danr/Papers/BrazAmRo08.pdf
87. Shafer G (1976) A mathematical theory of evidence. Princeton University Press, Princeton
88. Singla P, Domingos P (2005) Discriminative training of Markov logic networks. In: Proc of the natl conf on artificial intelligence
89. Smets P, Mamdani E, Dubois D, Prade H (eds) (1988) Non-standard logics for automated reasoning. Academic Press, San Diego

90. Stoilos G, Stamou G, Pan JZ, Tzouvaras V, Horrocks I (2007) Reasoning with very expressive fuzzy description logics. J Artif Intell Res 273–320
91. Straccia U (2001) Reasoning within fuzzy description logics. J Artif Intell Res 14:137–166
92. Straccia U (2006) A fuzzy description logic for the semantic web. In: Sanchez E (ed) Fuzzy logic and the semantic web, capturing intelligence. Elsevier, Amsterdam, pp 73–90. Chap 4
93. Straccia U (2008) Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In: Baroglio C, Bonatti PA, Maluszyński J, Marchiori M, Polleres A, Schaffert S (eds) Reasoning web. Springer, Berlin, pp 54–103
94. Straccia U, Bobillo F (2007) Mixed integer programming, general concept inclusions and fuzzy description logics. In: Proceedings of the 5th conference of the European society for fuzzy logic and technology (EUSFLAT-07), vol 2, University of Ostrava, Ostrava, Czech Republic, pp 213–220
95. Subrahmanian V (2007) Uncertainty in logic programming: some recollections. Assoc Log Program Newslett 20(2)
96. Subrahmanian VS (1987) On the semantics of quantitative logic programs. In: Proceedings of the 4th IEEE symposium on logic programming, pp 173–182
97. Valtorta M, Kim YG, Vomlel J (2002) Soft evidential update for probabilistic multiagent systems. Int J Approx Reason 29(1):71–106
98. Valtorta M, Dang J, Goradia H, Huang J, Huhns M (2005) Extending Heuer's analysis of competing hypotheses method to support complex decision analysis. In: Proceedings of the 2005 international conference on intelligence analysis (IA-05) (CD-ROM), extended version available at http://www.cse.sc.edu/~mgv/reports/IA-05.pdf
99. Zadeh LA (1978) Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst 1:3–28