Instantiation to Support the Integration of Logical and Probabilistic Knowledge

Jingsong Wang and Marco Valtorta

Department of Computer Science and Engineering
University of South Carolina, Columbia SC 29208, USA
{wang82, mgv}@cse.sc.edu

Abstract. Integrating the expressive power of first-order logic with the ability of probabilistic reasoning of Bayesian networks has attracted the interest of many researchers for decades. We present an approach to integration that translates logical knowledge into Bayesian networks and uses Bayesian network composition to build a uniform representation that supports both logical and probabilistic reasoning. In particular, we propose a new way of translation of logical knowledge, relation search, that is easy to understand, simple to implement, and efficient to execute. Grounding is required to generate a Bayesian network in the first-order case. In order to limit the size of the generated Bayesian networks, our prototype restricts the knowledge base to be function-free.

Keywords: Bayesian networks, First-order logic, Hybrid logical-probabilistic systems, Implicative normal form

1 Introduction

1.1 Automated Reasoning and the Integration Problem

Knowledge-based systems normally comprise two components: a knowledge base (KB) and an inference engine. The former encodes what we know about the world, while the latter acts on the knowledge base to answer queries [11, 2]. Traditionally, the knowledge bases consist of a set of logical rules, and the reasoning engine is based on logical deduction. Because of the unavoidable need of probabilistic reasoning, Bayesian networks (BNs) and the laws of probability theory play important roles in building knowledge-based systems.

However, a system based purely on logic or Bayesian networks is not practical for many advanced applications. Classical first-order logic (FOL) has great expressive power but cannot handle uncertainty. Bayesian networks can represent probabilistic information well but scale poorly and require users to have specialized expertise for effective modeling. This situation led many researchers to work on the integration problems of these two types of inference and various approaches have been proposed.

For the relation between logical reasoning and probabilistic reasoning, in case of propositional logic, logical reasoning is just one special case of probabilistic

reasoning. For FOL, whose reasoning is mostly based on instantiation and deduction (although there are also lifted approaches), it could be translated into propositional logic through instantiations (Herbrand Expansion), and therefore, in some sense, FOL could be considered a special case of probabilistic reasoning.

1.2 Using Logical Knowledge in Probabilistic Reasoning

Probabilistic models have been widely used in AI. Because of various data sources and ways of modeling, conflicts might appear in probabilistic reasoning of large systems, especially when comparing its result with commonsense conclusion from traditional logical reasoning. Purely probabilistic knowledge and modeling may not be accurate or strong enough, and we believe that the pre-existing logical knowledge could be a very important complement that should never be ignored in probabilistic reasoning. We can use a simple example to show the influence. Figure 1 shows a Bayesian network with two nodes A and B[1]. This Bayesian network represents our knowledge of probabilistic dependency of proposition A and proposition B. $\mathbf{Pr}(A)$ and $\mathbf{Pr}(B|A)$ quantify these dependencies. Then an abductive query for A's probability of being true given B is true is to calculate $\mathbf{Pr}(A|B)$, which has a unique value. In Figure 1, $\mathbf{Pr}(A = true|B = false) = 0.27$. However, suppose we also have some logical knowledge about the relation between A and B in KB, such as $A \rightarrow B$. Then there might be a disagreement with the posterior probability of A, as the logical knowledge requires us to conclude that A should be false, with 100% certainty. Which value should we choose? In some cases, we might like to accept the result from logic reasoning, as traditionally we have more experience with logical knowledge and we believe that logic reasoning is more mature in AI, while probabilistic knowledge is normally hard to quantify and therefore error-prone. This kind of case is possible especially in many design areas. The example shows that sometimes our knowledge might be inconsistent when modeling a problem from the logical view and the probabilistic view individually. Or we can say, the previous probabilistic model has not combined all our knowledge of A and B, and reasoning based on such a Bayesian network is insufficient.

If we use the logical relation to supplement the probabilistic model in a way as shown in Figure 2, i.e., we add an extra node representing the logical relationship, for the same evidence plus the additional logical knowledge, we get a different result of the posterior probability of A from the Bayesian network.

Things get more complex when we can have the uncertainty over logical rules from KB. However, the complexity also highlights the necessity of putting logical knowledge into consideration of probabilistic reasoning.

The previous example is very simple, as there are only two nodes in the original Bayesian network that are directly connected. For nodes with such direct connections, people should have better knowledge about their relations and the

[1] The probabilities shown in Figure 1 are computed using the commercial Bayesian network shell Hugin (www.hugin.com), which uses the junction tree algorithm for probabilistic inference [6].

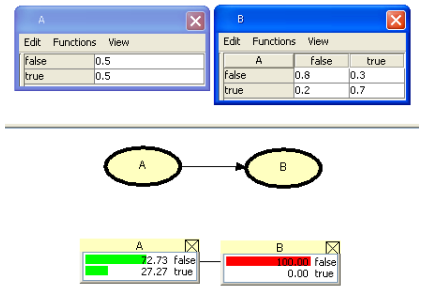

Fig. 1. A Bayesian network with its CPTs and reasoning result given evidence.

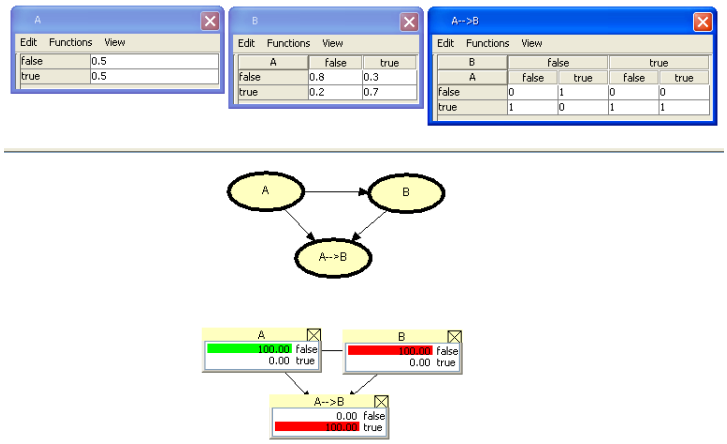

Fig. 2. Based on the Bayesian network in Figure 1, an extra node representing logical relations is added.

probabilistic dependency should be modeled consistently with the existing logical knowledge. However, we can think of a more complex example. We still have the same logic knowledge about A and B in the KB: $A \rightarrow B$. The original Bayesian network still contains node A and B, but there are also other nodes between them. Figure 3 shows a Bayesian network based on three nodes, one of which is the extra node C. Suppose now we are more interested in the posterior probability of C being true, given evidence B being true, which is shown to be is 0.14 in Figure 3. Now we integrate the logical relationship $A \rightarrow B$ into the new Bayesian network the same way as before, as shown in Figure 4. The posterior probability of C being true is 0.27, which is apparently increased, compared with the previous 0.14. Particularly, this shows the probabilistic influence of the logical relation of A and B to the proposition C even if there is no logical knowledge of C involved in the KB.

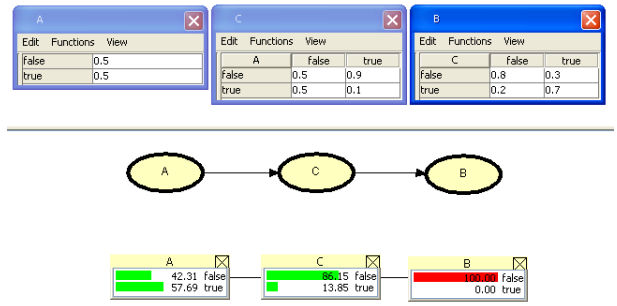

Fig. 3. A Bayesian network with its CPTs and reasoning result given evidence.

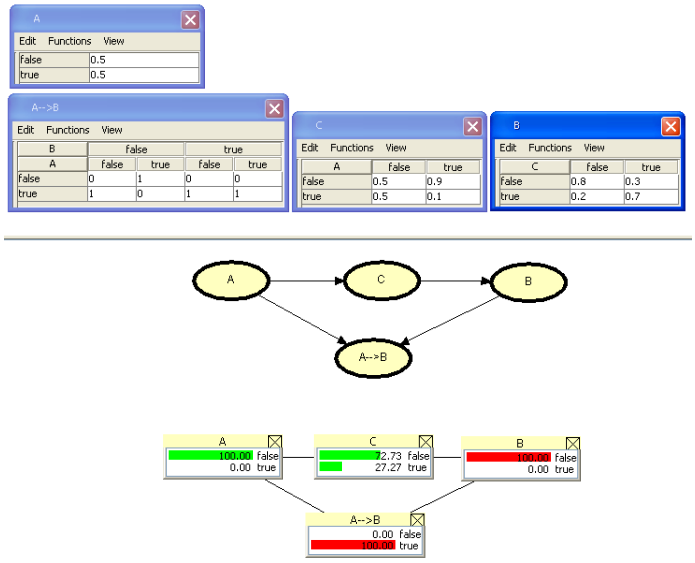

Fig. 4. An extra logical node is added to the Bayesian network in Figure 3.

This way of combining logical knowledge into probabilistic reasoning could be more meaningful when the original Bayesian networks are more complex. For example, just assume that nodes A and B in Figure 3 are now connected through ten other nodes. Note that the connection is not necessarily a line, but a network. In this case, because the distance from A to B is too long, people cannot ensure that, when they are modeling the probabilistic dependencies using Bayesian networks, they have put into enough consideration logical knowledge, which could have been gained from an existing KB. Under this situation, a simple

logical partial model, such as the ones in Figure 2 and Figure 4, can achieve this easily.

A special case is that when we have no knowledge over the logical rules, i.e., no evidence over logical rules, the composite model just resumes to the basic probabilistic model. This is because of the way the logical model is generated: it is a two level Bayesian network (in propositional case) where d-separations will appear when no evidence over logical rule nodes, which always serve as children of atomic formula nodes.

1.3 Our Framework for Integration

We pursue an approach that allows modelers to 1) specify special knowledge using the most suitable languages, while reasoning in an uniform engine; 2) make use of pre-existing logical knowledge bases for probabilistic reasoning (to complete the model or minimize potential inconsistencies).

In our framework, the user is assumed to use FOL formulas to specify logical knowledge stored in a KB, and a Bayesian network, which we call *Probabilistic Bayesian Network (PBN)*, to specify probabilistic knowledge. The basic idea is to convert related knowledge from the logical KB into a logical model represented through a Bayesian network, which we call *Logical Bayesian Network (LBN)*. Then the LBN is composed with the PBN. The final output of the composition is a Bayesian network, *Composite Bayesian Network (CBN)*, that integrates both the logical knowledge and the probabilistic knowledge related to the query. Figure 5 summarizes the process. Note that we still follow the knowledge-based approach for reasoning, but our knowledge base is neither only the traditional logical knowledge base nor simply the Bayesian networks, but their combination. However reasoning is still based on the laws of probability theory.

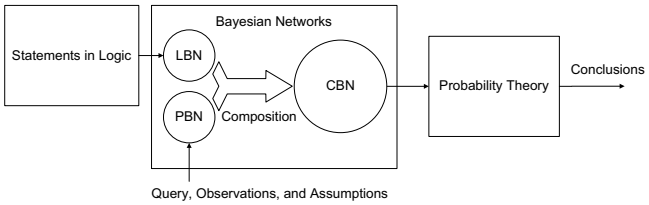

Fig. 5. Reasoning systems with the combination of logic and Bayesian networks.

The main focus of this paper is the way the logical model is generated. More details about the composition process and a comparison with related work can be found in [12]. Here we improve the work of [12] by removing the need for a theorem prover. The remainder of this paper is organized as follows. In section 2, we introduce the relation search algorithm in the propositional case at first

and then we extend it into the FOL case. A proof of correctness follows and an instantiation problem is discussed. Then we conclude this paper in section 3.

2 Logical Bayesian Network Generation

2.1 Propositional Case Algorithm

The end purpose is to build a composite model from logical knowledge and probabilistic knowledge. The PBN contains most of probabilistic knowledge, the nodes of which consist of the query, observations, and assumptions. We call them *probabilistic atoms*. When there is no direct probabilistic dependency specified among probabilistic atoms, the Bayesian network contains a set of isolated nodes.

The logical knowledge is assumed to be from a logical KB comprising a set of logical formulas. We will translate the KB directly into a LBN through a search algorithm, called *relation search*. The basic idea of relation search is very simple. We just look through the KB and extract all the logical formulas that are related to the probabilistic atoms. A formula has a *relation* with an atom if that atom appears directly or indirectly in this formula. The indirect appearance is defined the following way. If atom A_1 and A_2 both appear in a formula R_1, and A_1 also appears in a formula R_2 while A_2 does not, we say that A_2 appears indirectly in the formula R_2. Algorithm 1 depicts the pseudocode for our relation search algorithm.

Note that in Algorithm 1, we only explicitly generate the CPTs for the formulas from KB. These CPTs are determined by the formulas' logical structure. For example, for formula $A \vee B$, the value true has probability 1 if and only if either A or B is true. $BN(\mathcal{V}, \mathcal{E}, \mathcal{L})$ denotes the Bayesian network built based on $(\mathcal{V}, \mathcal{E}, \mathcal{L})$, where $\mathcal{V}$ represents the set of nodes (variables), $\mathcal{E}$ represents the set of directed edges that connect nodes together, and $\mathcal{L}$ represents the set of conditional probability tables.

2.2 FOL Case Modification of Algorithm

In the FOL case, instantiation and quantified formula node generation are needed to build the LBN. Even in the FOL case, queries, observations and assumptions should mostly be ground formulas. Thus we only consider ground atomic formulas from PBN and the set of such atoms is denoted by $\mathcal{P}$. We assume that all the formulas in KB are closed (i.e., no occurrence of free variables) and in Skolem form, which allows only universal quantifiers. Also, we assume that constants in KB are all the individuals in the domain, and we restrict the domain to be function-free.

1. We do the same search and find all the related formulas $\mathcal{R}$ from KB.
2. We use the available constants in $\mathcal{R} \cup \mathcal{P}$ to get all the possible ground instantiations of quantified formulas in $\mathcal{R}$, and add these instantiations to $\mathcal{R}$. This new set is named $\mathcal{R}'$.

Algorithm 1 Relation Search

Require: the probabilistic atom set $\mathcal{P}$ from PBN containing query, observations, and assumptions, the set of logical formulas KB $= \{R_1, R_2, ..., R_z\}$, $z = |\text{KB}|$, and the sets $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_z$. $\mathcal{A}_i$ comprises all the atoms appearing in its corresponding logical formula $R_i \in$ KB, where $1 \leq i \leq z$.

$\mathcal{V} = \mathcal{P}$;
$\mathcal{E} = \emptyset$;
$\mathcal{L} = \emptyset$;
for $i = 1$ to z **do**
  Tag R_i as not visited;
end for
while true do
  $Changed \Leftarrow$ **false**;
  for $i = 1$ to z **do**
    if R_i is not visited **then**
      if $\mathcal{A}_i \cap \mathcal{V} \neq \emptyset$ **then**
        $\mathcal{V} = \mathcal{V} \cup \mathcal{A}_i \cup \{R_i\}$;
        for all $A \in \mathcal{A}_i$ **do**
          $\mathcal{E} = \mathcal{E} \cup \{(A, R_i)\}$;
        end for
        Build CPT Θ_i for R_i based on its logical structure;
        $\mathcal{L} = \mathcal{L} \cup \{\Theta_i\}$;
        Tag R_i as visited;
        $Changed \Leftarrow$ **true**;
      end if
    end if
  end for
  if $Changed =$ **false then**
    break;
  end if
end while
return $BN(\mathcal{V}, \mathcal{E}, \mathcal{L})$;

3. We build a Bayesian network based on $\mathcal{R}'$. We follow exactly the same procedure as in the propositional case for generating nodes, edges, and CPTs for the propositional formulas in $\mathcal{R}'$. For a quantified formula, we put it as the child of its ground instantiations (groundings) plus an extra node O, which represents a proposition that all the other instantiations that are based on constants appearing in KB but not in $\mathcal{R}'$ hold. The CPT for such a quantified formula is an AND table, i.e., the value true has probability 1 if all parents are true and probability 0 otherwise.

Thus for the FOL case, the generated Bayesian network will usually be a three level network if quantified formulas exist in $\mathcal{R}'$ [2]. The nodes corresponding

[2] One extreme case is that the quantified formula itself is an atomic formula. The parents of the corresponding node are root nodes in **G** directly. However, such formulas do not make sense in a normal KB. We exclude this case.

to them are in the third level. One important change for the FOL case relation search output is an O node for each quantified formula as one additional parent, whose value reflects the actual influence of some constants, which seems to be unrelated, through the related formulas.

Notice that any instantiation of the root nodes (including O nodes) of the resulting Bayesian network has defined one truth value for any formula appearing in the Bayesian network, as the Bayesian network has explicitly or implicitly contained all the constants from KB and we have assumed that the constants available in KB have been all the constants in the domain. For a ground atomic formula (including O node), which appears as the root node, its truth value is directly decided by its state value in the instantiation. For a ground compound formula, its truth value is decided by its logical evaluation based on its parents' truth values. For a quantified formula, its truth value is decided by the logical AND evaluation based on its parents' truth values.

We can use one example to illustrate how instantiation works. Suppose we have a logical KB containing formulas as shown in Table 1. $\mathcal{P} = \{P(a), P(b)\}$ are from PBN [3]. Then after the relation search,

$$\mathcal{R} = \{\forall x P(x) \rightarrow Q(x)\}.$$

Here constants are a, b, and c. We note that the second and the third formula in KB are ignored as they cannot be related. The output Bayesian network $\mathbf{G}$ is shown in Figure 6.

Table 1. The original FOL KB's formulas.

1 $\forall x P(x) \rightarrow Q(x)$,
2 $\forall x H(x) \rightarrow L(x)$,
3 $H(c)$.

2.3 Correctness

We want to show that models constructed in relation search behave according to our logical intuitions.

Theorem 1. *For a BN resulting from relation search,* $\mathbf{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$*, for any* $U \in \mathcal{V}$ *and any* $\mathcal{V}' \subseteq \mathcal{V}$*, if* $\mathcal{V}' \models U$*, then* $\mathbf{Pr}(U = true | \mathcal{V}' = true) = 1$ *in* $\mathbf{G}$*, where* $\mathcal{V}' = true$ *means that all the nodes in* $\mathcal{V}'$ *of* $\mathbf{G}$ *are set to true.*

Proof of Theorem 1 The idea is to use weighted model counting [2], over the Bayesian network $\mathbf{G}$, to conclude the joint probability of evidence $\mathbf{e} = \{U = false\} \cup \mathcal{V}' = true$ to be 0, and then follow Bayes' rule to conclude the posterior

[3] Note that the PBN can be the result of any modeling technique, such as MEBN [9, 10] or BLP [7], that outputs a Bayesian network.

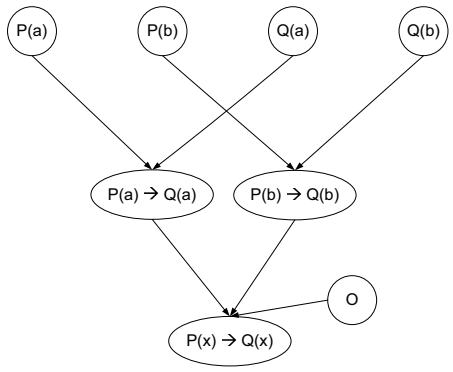

Fig. 6. A FOL case LBN example.

probability $\mathbf{Pr}(U = true|\mathcal{V}' = true) = 1$. We can assume $\mathcal{V}' = \{S_1, S_2, ..., S_k\}$, where $k \leq |\mathcal{V}|$. Then evidence can be represented by $\mathbf{e} = \{\overline{u}, s_1, s_2, ..., s_k\}$.

If $\mathcal{V}' \models U$, then FOL theory $\{\neg U\} \cup \mathcal{V}' = \{\neg U, S_1, S_2, ..., S_k\}$ is logically unsatisfiable. We define $F = U \wedge S_1 \wedge S_2 \wedge ... \wedge S_k$ and $F' = \neg U \wedge S_1 \wedge S_2 \wedge ... \wedge S_k$. Then a Herbrand structure of F is also the one of F' and vice versa. Note that every subformula of F has a corresponding node in $\mathbf{G}$. We know that F' is unsatisfiable. This means that all interpretations (more commonly called structures in FOL) of F', including its Herbrand structures, that evaluate formula U to false will also evaluate at least one formula S_x $(1 \leq x \leq k)$ to false.[4]

For BN $\mathbf{G}$, each instantiation includes one instantiation of $\mathbf{G}$'s root nodes, which are all ground atomic formulas because of the way $\mathbf{G}$ is built. (The root nodes also include the O nodes, which are also atomic propositions.) Such an instantiation of root nodes in $\mathbf{G}$ has defined a Herbrand structure for the language of $\mathbf{G}$. Thus given an instantiation $\mathbf{G}$, there is a corresponding evaluation for any formula appearing in $\mathbf{G}$ and this evaluation result is fixed. For example, the ground logical constituents of nodes $\{U, S_1, S_2, ..., S_k\} \subseteq \mathcal{V}$ are root nodes in $\mathbf{G}$. Each instantiation of the root nodes of $\mathbf{G}$ contains one Herbrand structure for formula $F = U \wedge S_1 \wedge S_2 \wedge ... \wedge S_k$ and its subformulas.

In addition, for each quantified formula appearing in $\mathbf{G}$, all its possible groundings based on constants available in the relation search result $(\mathcal{R} \cup \mathcal{P})$ also appear in $\mathbf{G}$.

Consider the special CNF encoding introduced in Section 11.6.1 [2] of BN $\mathbf{G}$. Any instantiation of $\mathbf{G}$ corresponds to a model of such a CNF encoding. To compute $\mathbf{Pr(e)}$ through the weighted model count, we only consider instantiations of $\mathbf{G}$ that are compatible with evidence $\mathbf{e} = \{\overline{u}, s_1, s_2, ..., s_k\}$.

[4] For a closed formula in Skolem form, it is satisfiable if and only if it has a Herbrand model. In other words, unsatisfiability ≡ no Herbrand model.

We choose one arbitrary instantiation of $\mathbf{G}$ compatible with $\mathbf{e}$. The Herbrand structure contained in the current instantiation is denoted by H. We know that formula U can evaluate to true or false under H. We consider these two cases separately.

1. U evaluates to false in H.
   Thus U is false in H. Because of the unsatisfiability of F', we know that F' does not have a Herbrand model. Therefore if U evaluates to false in H, there must be at least a S_x that evaluates to false in H. As we know that the nodes of $\mathbf{G}$ are in three levels, node S_x might appear in the first level (being an atomic formula node), in the second level (being a ground formula), or in the third level (being a quantified formula). Note that atomic formulas, appearing as root nodes in $\mathbf{G}$, definitely evaluate to the same values as their state values in the instantiation. Remember that all the nodes $S_1, S_2, ..., S_k$ are in state true, as the current instantiation is compatible with evidence $\mathbf{e}$. Since S_x is false in H, S_x could not be in the first level. This also excludes the case that S_x is an O node. If S_x is in the second level, then there is an inconsistency between its logical evaluation, which is false based on its parents' state values, and its state value, which is true as determined by $\mathbf{e}$. If node S_x is in the third level, i.e., it is a quantified formula, there are two cases for possible instantiations of its parent nodes, which are groundings of S_x, in $\mathbf{G}$:
   (a) All the parent nodes of S_x are in state value true.
   Consider the Herbrand structure H for F contained in the current instantiation. Based on the definition of the truth value of quantified formulas with universal quantifier, because H evaluates S_x (which has universal quantifier) to false, there must be at least one grounding of S_x, denoted by S_{xg}, that evaluates to false in H. Based on the way $\mathbf{G}$ is built, S_{xg} is a parent node of S_x and it is in the second level of $\mathbf{G}$. (Note that the O node associated with S_x is excluded from being S_{xg}, because it is a root node, which should have the same evaluation in H as its state value, and a parent of S_x, and we have assumed that S_x's parent nodes are all in state value true) However, we have assumed that S_x's parent nodes are all in state value true. Thus for S_{xg}, there is a similar inconsistency as before between its logical evaluation, which is false based on its parents' state values, and its state value, which is true.
   (b) There is at least one parent node of S_x in state false, including its O node.
   The CPT for node S_x, which represents a formula with universal quantifier, is an AND table. There is still a similar inconsistency as discussed before between S_x's logical evaluation, which is false based on its parents' state values, and its real state value, which is true in order to be compatible with $\mathbf{e}$.
   Therefore, if node S_x is in the third level of $\mathbf{G}$, for any possible instantiation compatible with $\mathbf{e}$, we can always find a node with logical inconsistency. This node might be S_x itself, or one of its parents S_{xg}.

2. U evaluates to true in H.
   There are three cases for the appearance of U: U is an O node, a ground formula that is not an O node, and a quantified formula.
   (a) U is an O node. This case is impossible.
   Being a root node, an O node should have the same evaluation as its state value false, which is compatible with evidence $\mathbf{e}$.
   (b) U is a ground formula but not an O node.
   Still, U cannot be a root node. Thus U is a second level grounding. U has a state value false, which is compatible with evidence $\mathbf{e}$, and however it evaluates to true. Thus U itself has an inconsistency.
   (c) U is a quantified formula. We have a similar analysis as before.
   If all the parent nodes of U are in state value true, then there is an inconsistency in U, as we know that the CPT of U is an AND table, and however its state value is false to be compatible with $\mathbf{e}$.
   Otherwise, there is at least one parent in state value false. Based on the definition of the truth value of a quantified formula, all the parent formulas (node O or not) of U in $\mathbf{G}$ need to evaluate to true in H. Being a root node, the O node cannot be in state value false, as its state value should be the same value as its evaluation in H. Therefore there must be a grounding in state value false, which however evaluates to true. Then this grounding has an inconsistency.

Because of the way CPTs are built in $\mathbf{G}$, as shown in the propositional case analysis, the entries of a CPT of a non-root node in $\mathbf{G}$ all have value 0, if the logical evaluation of its logical formula, given its parents' state values, is inconsistent with the node's real state value. For a node with such inconsistency, denoted by S, if we use c to represent the state values of its parent nodes in the current instantiation, and the corresponding truth assignment of CNF encoding is denoted by ω, we know that

$$Wt(P_{s|c}) = \theta_{s|c} = 0,$$

and the weight of the model ω is

$$Wt(\omega) = Wt(P_{s|c}) * \prod_{i=1}^{t-1} Wt(P_i) = 0,$$

where t is the number of nodes in $\mathbf{G}$, i.e., $|\mathbf{G}| = t$, and P_i represents the other parameter literal of the model ω that is not $P_{s|c}$.

This conclusion holds for all the other models. Thus

$$\mathbf{Pr}(\mathbf{e}) = \mathbf{Pr}(\overline{u}, \mathcal{V}' = true) = \mathbf{Pr}(\overline{u}, s_1, s_2, ..., s_k) = \sum_{i=1}^{2^{t-k-1}} Wt(\omega_i) = 0,$$

where ω_i represents the model corresponding to the different instantiation of $\mathbf{G}$ compatible with evidence $\{\overline{u}, s_1, s_2, ..., s_k\}$. Thus, as claimed in the statement of the theorem,

$$\mathbf{Pr}(u|\mathcal{V}' = true) = \frac{\mathbf{Pr}(u, \mathcal{V}' = true)}{\mathbf{Pr}(u, \mathcal{V}' = true) + \mathbf{Pr}(\overline{u}, \mathcal{V}' = true)} = \frac{\mathbf{Pr}(u, \mathcal{V}' = true)}{\mathbf{Pr}(u, \mathcal{V}' = true) + 0} = 1.$$

2.4 A Partial Instantiation

In the FOL case, as the number of atoms containing different variables in a quantified formula increases, the number of groundings could increase exponentially. This might make the BN **G** very large. However, many nodes, including atoms and groundings, in this BN are not meaningful in real use.

In a well-defined KB, besides formulas representing rules, there are some facts that represent basic and fixed features of a few context constants in KB. For atoms describing such features, any groundings of them that are not explicitly specified in KB are regarded impossible. This is often referred as closed world assumption, i.e., for a feature F of a sequence of constants $\mathcal{C}$ in the domain, if $F(\mathcal{C})$ is not listed as a fact, then we believe $F(\mathcal{C})$ is false. We call the predicates like F *context predicates* and the set of related facts *context facts*.[5] For example, consider two atoms $Person(a)$ and $Table(b)$ as existing facts in KB. For the constants specified by these two atoms, their features are not exchangeable, as we know that $Person(b)$ and $Table(a)$ are meaningless. There should not be uncertainty over them any time as they are always false. Therefore, we can control the number of groundings of quantified formulas by discarding meaningless instantiations. In addition, we assume that all the rules in KB are in Implicative Normal Form (INF). Note that any formula can be easily translated into INF. For example, $(A_1 \wedge A_2) \vee (B_1 \wedge B_2) \rightarrow C$, where A_1, A_2, B_1, and B_2 are literals and C is a subformula, can be converted into $A_1 \wedge A_2 \rightarrow C$ and $B_1 \wedge B_2 \rightarrow C$. Then we add one extra step for the instantiation process of quantified formulas, which are in INF. We will still try all the possible instantiations of a quantified formula that contains context predicates. We discard a grounding unless all of the context predicates appearing in its body are context facts. The idea is simple. Each rule in KB naturally adds one constraint to the set of possible worlds. However, when one part of the body of the rule, which is in INF, is deterministically false, the rule will not constrain anything, based on the definition of logical implication. Then adding the rule is truly meaningless. Thus we can just ignore such groundings when building the BN. In applications, the users can define their context predicates and context facts flexibly for controlling the size of resulting BN.

3 Conclusion

In this paper, we presented a new way of translating logical knowledge into Bayesian networks that supports a new approach to the integration problem of logical and probabilistic reasoning that is easy to understand, simple to implement, and efficient to execute.

The method presented in this paper can be used not only for the general integration problem but also the traditional BN learning problem. There have been

[5] *Context predicates* and *context facts* play similar roles in instantiation as *random variable declarations* in LBN [4] and *context terms* in MEBN [9, 10]; these are all similar to typed predicates and terms.

classical ways to learn BNs, which include parameter estimation and structure learning, such as the EM algorithm [3] for parameter estimation with incomplete data and score-based learning for BN structure [1]. Meanwhile, Inductive logic programming (ILP), which is concerned with relational data mining, has been greatly researched. The problem of learning logic programs is the first and still most commonly addressed problem in ILP [5]. Because our approach provides a way of translating logic programs into Bayesian networks, we can easily extend most existing ILP techniques to work on BN learning.

We also notice that one quantitive label of a formula could be produced from the logic program's learning process. This label represents, e.g., the formula's rate of successful matching in the training data. Thus it is ideal to be used as soft evidence [8] in the composite model BN resulting from the integration process for further probabilistic reasoning. How to use such soft evidence for the improvement of accurate reasoning for many problems, such as classification, is an interesting topic for future work .

References

1. Cooper, G.F., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9, 309–347 (1992)
2. Darwiche, A.: Modeling and Reasoning with Bayesian Networks. Cambridge University Press (2009)
3. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society, Series B 39(1), 1–38 (1977)
4. Fierens, D., Blockeel, H., Bruynooghe, M., Ramon, J.: Logical bayesian networks and their relation to other probabilistic logical models. In: Proceedings of the 15th International Conference on Inductive Logic Programming. pp. 121–135. Springer (2005)
5. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press (2007)
6. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs, second edition. Springer, New York, NY (2007)
7. Kersting, K., Raedt, L.D.: Bayesian logic programs. CoRR cs.AI/0111058 (2001)
8. Kim, Y.G., Valtorta, M., Vomlel, J.: A prototypical system for soft evidential update. Applied Intelligence 21(1) (July–August 2004)
9. Laskey, K.B.: First-order Bayesian logic, Technical Report C4I06-01. Tech. rep., SEOR Department, George Mason University (February 2006)
10. Laskey, K.B.: MEBN: A language for first-order knowledge bases. Artificial Intelligence 172, 140–178 (2008)
11. McCarthy, J.: Programs with common sense. In: Semantic Information Processing. pp. 403–418. MIT Press (1959)
12. Wang, J., Byrnes, J., Valtorta, M., Huhns, M.: On the combination of logical and probabilistic models for information analysis. Applied Intelligence pp. 1–26 (January 2011)