ELSEVIER Theoretical Computer Science 171 (1997) 147–177

Theoretical Computer Science

Answering queries from context-sensitive probabilistic knowledge bases[1]

Liem Ngo, Peter Haddawy *

Decision Systems and Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Wisconsin-Milwaukee, Milwaukee, WI 53201, USA

Abstract

We define a language for representing context-sensitive probabilistic knowledge. A knowledge base consists of a set of universally quantified probability sentences that include context constraints, which allow inference to be focused on only the relevant portions of the probabilistic knowledge. We provide a declarative semantics for our language. We present a query answering procedure that takes a query Q and a set of evidence E and constructs a Bayesian network to compute $P(Q|E)$. The posterior probability is then computed using any of a number of Bayesian network inference algorithms. We use the declarative semantics to prove the query procedure sound and complete. We use concepts from logic programming to justify our approach.

Keywords: Reasoning under uncertainty; Bayesian networks; Probability model construction; Logic programming

1. Introduction

Bayesian networks [23] have become the most popular method for representing and reasoning with probabilistic information [14]. But the complexity of inference in Bayesian networks [5] limits the size of the models that can be effectively reasoned over. Recently, the approach known as knowledge-based model construction [29] has attempted to address this limitation by representing probabilistic information in a knowledge base using schematic variables and indexing schemes and then constructing a network model tailored to each specific problem. The constructed network is a subset of the domain model represented by the collection of sentences in the knowledge base. Approaches to achieving this functionality have either focused on practical representations and model construction algorithms, neglecting formal aspects of the problem [11, 4], or they have focused on formal aspects of the knowl-

* Corresponding author. E-mail: {liem, haddawy}@cs.uwm.edu.

[1] This work was partially supported by NSF grants IRI-9207262, IRI-9509165, and a Graduate School Fellowship from the University of Wisconsin-Milwaukee.

edge base representation language, with less concern for ease of use of the representation and practical algorithms for constructing networks [24, 2]. Hence, none of this work has provided a practical network construction algorithm which is proven correct.

We take a more encompassing approach to the problem by presenting a natural representation language with a rigorous semantics together with a network construction algorithm. In previous work [12] we started to address this problem by presenting a framework for constructing Bayesian networks from knowledge bases of first-order probability logic sentences. But the language used in that work was too constrained to be applicable to many practical problems. The current paper makes three main contributions. First, we define a logical language that allows probabilistic information to be represented in a natural way and permits the probabilistic information to be qualified by context information, when available. A knowledge base consists of a set of universally quantified sentences of the form $(P(\text{consequent} \mid \text{antecedent}) = \alpha) \leftarrow \text{context}$. Second, we provide a declarative semantics for the language. Third, we present a sound and complete query answering procedure that uses a form of SLD resolution to construct Bayesian networks and then uses the networks to compute posterior probabilities. Throughout our development we highlight the relationship between our framework and both logic programming and Bayesian networks.

When reasoning with a large knowledge base, context information is necessary in order to focus attention on only the relevant portions. Consider the following motivating example, which will be referred to throughout the remainder of the paper. A burglary alarm could be triggered by a burglary, an earthquake, or a tornado. The likelihood of a burglary is influenced by the type of neighborhood one resides in. In different contexts, people might have different beliefs concerning the probabilistic relations between these variables. Someone who resides in California might think of an earthquake when he hears an alarm. If he lived in Wisconsin instead, he might relate that event with a possible tornado. Two networks representing these two different contexts are shown in Fig. 1. Many problems, like this one, will naturally contain some deterministic information that can be used as context to index relevant probabilistic relations. For simplicity of exposition, we have chosen this toy problem to use as a running example to illustrative the details of the formalism. A discussion of some practical problems in which such deterministic information can be identified is provided in Section 6.

2. Representation language

Throughout this paper, we use P and sometimes P^* to denote a probability distribution; $A, B, \ldots$ with possible subscripts to denote atoms; names starting with a capital letter or X to denote domain variables; and $p, q, \ldots$ with possible subscripts to denote predicates. We distinguish two types of predicates: *context and probabilistic predicates.*

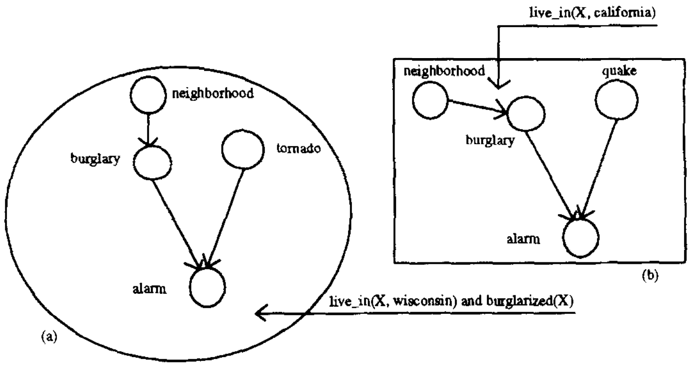

Fig. 1. An example of context-dependent beliefs.

Context predicates (c-predicates) have value true or false and are deterministic. They are used to describe the context the agent is in and to eliminate unnecessary or inappropriate probabilistic information from consideration by the agent. An atom formed from a context predicate is called a *context atom* (*c-atom*). A *context literal* (*c-literal*) is either a c-atom or the negation of a c-atom. A *context base* is a normal logic program [17], which is a set of universally quantified sentences of the form $C_0 \leftarrow L_1, L_2, \ldots, L_n$, where n is a nonnegative integer, $\leftarrow$ stands for implication, comma for logical conjunction, C_0 for a c-atom, and $L_i, 1 \leqslant i \leqslant n$, for c-literals. We use *completed logic programs* proposed by Clark [17] for the semantics of the context base. Essentially, a completed logic program is formed from a logic program by adding to it (1) some clauses so that the clauses in the original program become the only definitions of the predicates in the program, and (2) the *equality theory* to interpret the introduced equality predicate [17].

Each probabilistic predicate (p-predicate) represents a class of similar random variables. P-predicates appear in probabilistic sentences and are the focus of probabilistic inference processes. An atom formed from a p-predicate is called a probabilistic atom (p-atom). Queries and evidence are expressed as p-atoms. In the probability models we consider, each random variable can assume a value from a finite set and in each possible realization of the world, that variable can have one and only one value. We capture this property by requiring that each p-predicate have at least one attribute, which represents the value of the corresponding random variable. For predicates with multiple attributes, the value attribute is always the last. For example, the random variable *neighborhood* of a person x can have value *bad*, *average*, *good* and can be represented by a two-position predicate – the first position indicates the person and the second indicates the type of that person's neighborhood. A p-predicate with multiple attributes represents a set of related random variables – one for each ground instance of the non-value attributes. Associated with each p-predicate p must be a statement of the form $VAL(p) = \{v_1, \ldots, v_n\}$, where $v_1, \ldots, v_n$ are constants denoting the possible values of the corresponding random variables.

Definition 1. Let $A = p(t_1,\dots,t_{m-1},t_m)$ be a p-atom, we use $obj(A)$ to designate the tuple $(p,t_1,\dots,t_{m-1})$ and $val(A)$ to designate t_m. If A is a p-atom of predicate p, then $VAL(A)$ means $VAL(p)$. If A is a ground p-atom then $obj(A)$ represents a concrete random variable in the model and $val(A)$ is its value. We also define $Ext(A)$, *the extension of* A, to be the set $\{p(t_1,\dots,t_{m-1},v_i) \mid 1 \leqslant i \leqslant n\}$.

If A is a ground p-atom, we assume that the elements of $Ext(A)$ are mutually exclusive and exhaustive: in each "possible world", one and only one element in $Ext(A)$ is true. We assume that the attributes of p-predicates are typed, so that each attribute is assigned values in some well-defined domain.

Example 2. In the context of our running example, the declarations $nbrhd(Somebody : Person, V)$ and $VAL(nbrhd) = \{bad, average, good\}$ say that the first attribute of *nbrhd* is a value in the domain *Person*, which is the set of all persons, and the second attribute can take values in the set $\{bad, average, good\}$. For a person, say named John, $nbrhd(john, good)$ means that the random variable "neighborhood of John", indicated in the language by $obj(nbrhd(john, good))$, is "good", indicated in the language by $val(nbrhd(john, good)) = good$. In any possible world, one and only one of the following atoms is true: $nbrhd(john, bad)$, $nbrhd(john, average)$, and $nbrhd(john, good)$.

We denote the set of all such predicate declarations in a knowledge base by *PD*.

A *probabilistic sentence* (*p-sentence*) has the form $(P(A_0|A_1,\dots,A_n) = \alpha) \leftarrow L_1,\dots,L_m$ where $n \geqslant 0$, $m \geqslant 0$, $0 \leqslant \alpha \leqslant 1$, A_i are p-atoms, and L_j are c-literals. The sentence can have free variables and each free variable is universally quantified over the entire scope of the sentence. The above sentence is called context-free if $m = 0$. If S is the above probabilistic sentence, we define $context(S)$ to be the conjunction $L_1 \wedge \cdots \wedge L_m$, $prob(S)$ to be the probabilistic statement $P(A_0|A_1,\dots,A_n) = \alpha$, $ante(S)$ to be the conjunction $A_1 \wedge \cdots \wedge A_n$, and $cons(S)$ to be A_0. For convenience, sometimes we take $ante(S)$ to be the set of the conjuncts. We call A_0 the consequent of S. A probabilistic base (*PB*) of a knowledge base is a finite set of p-sentences.

Definition 3. Let A be a (p- or c-) atom. We define $ground(A)$ to be the set of all ground instances of A satisfying the domain (type) constraints. If E is a set of p-atoms then $ground(E) = \bigcup \{ground(A) \mid A \in E\}$. Let S be a p-sentence. We define $ground(S)$ to be the set of all ground instances of S satisfying the domain constraints.

A set of ground p-atoms $\{A_i \mid 1 \leqslant i \leqslant n\}$ is called *coherent* if $\forall i,j \ (1 \leqslant i,j \leqslant n) : obj(A_i) = obj(A_j) \rightarrow val(A_i) = val(A_j)$.

A *PB* will typically not be a complete specification of a probability distribution over the random variables represented by the p-atoms. One type of information that may be lacking is the specification of the probability of a variable given combinations of values of two or more variables that influence it. For real-world applications, this type of information can be difficult to obtain. For example, for two diseases D_1 and D_2 and a symptom S we may know $P(S|D_1)$ and $P(S|D_2)$ but not $P(S|D_1,D_2)$. Combining rules such as generalized noisy-OR [6, 26] are commonly used to construct such combined influences.

We define a *combining rule* as any algorithm that

(i) takes as input a set of *ground* p-sentences that have the same consequent $\{Pr(A_0s|A_{i1},\dots,A_{in_i}) = \alpha_i \mid 1 \leqslant i \leqslant m\}$ where m is a non-negative integer or infinity, and

(ii) produces as output a set of ground p-sentences $OS = \{Pr(A_0|B_{i1},\dots,B_{ik_i}) = \beta_i \mid 1 \leqslant i \leqslant h,\ h$ and k_i are some non-negative finite integers$\}$ such that

(a) the antecedent of each p-sentence in OS is a coherent set,

(b) if S_1 and S_2 are two different p-sentences in OS then $ante(S_1) \cup ante(S_2)$ is not coherent, and

(c) if B appears in the antecedent of a p-sentence in OS then $obj(B) \in \bigcup_{i=1}^{m}\{obj(A_{i1}),\dots,obj(A_{in_i})\}$; and

(iii) the output set is empty if and only if the input set is empty.

Condition (b) enforces the constraint that at most one antecedent is true in any possible world. As a result, in each possible world there is at most one statement about the direct influences on the consequent A_0. Condition (c) is natural: we do not want to introduce new random variables.

We assume that for each p-predicate p, there exists a corresponding combining rule in CR, the combining rules component of the *KB*. Each combining rule is applied only once for any given predicate. The combining rules are specified by the designer of a knowledge base and they reflect his conception of the interaction between direct influences of the same random variable.

A knowledge base (*KB*) consists of predicate descriptions, a probabilistic base, a context base, and a set of combining rules. We write $KB = \langle PD, PB, CB, CR\rangle$.

Fig. 2 shows a possible knowledge base for representing the burglary example introduced in the previous section. We have the following p-predicates: *nbrhd*, *burglary*, *quake*, *alarm*, and *tornado*. In Example 2 we gave the interpretation of the statements $nbrhd(Somebody : Person, V)$ and $VAL(nbrhd) = \{bad, average, good\}$. Similar interpretations apply to other statements in *PD*.

PB contains the p-sentences. Due to space limitations, we cannot show all the sentences. We would have sentences describing the conditional probabilities that the random variables corresponding to *burglary* and *alarm* achieve the value *no*, and the marginal probability for *quake* achieving the value *no*. *CB* specifies relationships in the context information. For example, the first two clauses, define the predicate *live_in*: if a person X lives in the area Y then she lives in Y; if she lives in Y and Y is in Z then she also lives in Z. The last clause in *CB* declares that Madison is in Wisconsin. Most of the probabilistic sentences in *PB* are annotated by the corresponding context. For example, if a person lives in California then the chance that her neighborhood is bad is 0.3. The $\neg$ operator is *negation-as-failure* used in logic programming to encode

$PD = \{$ $nbrhd(Somebody : Person, V), VAL(nbrhd) = \{bad, average, good\};$
$burglary(Somebody : Person, V), VAL(burglary) = \{yes, no\};$
$quake(Somearea : Area, V), VAL(quake) = \{yes, no\};$
$alarm(Somebody : Person, V), VAL(alarm) = \{yes, no\};$
$tornado(Somearea : Area, V), VAL(tornado) = \{yes, no\}\}$

$PB = \{$

(S_1) $P(nbrhd(X, average)) = .4$
(S_2) $P(nbrhd(X, bad)) = .3 \leftarrow live\_in(X, california)$
(S_3) $P(nbrhd(X, good)) = .3 \leftarrow live\_in(X, california)$
(S_4) $P(nbrhd(X, bad)) = .2 \leftarrow live\_in(X, wisconsin)$
(S_5) $P(nbrhd(X, good)) = .4 \leftarrow live\_in(X, wisconsin)$
(S_6) $P(burglary(X, yes)|nbrhd(X, average)) = .4$
(S_7) $P(burglary(X, yes)|nbrhd(X, good)) = .3 \leftarrow live\_in(X, california)$
(S_8) $P(burglary(X, yes)|nbrhd(X, bad)) = .6 \leftarrow live\_in(X, california)$
(S_9) $P(burglary(X, yes)|nbrhd(X, good)) = .2 \leftarrow live\_in(X, wisconsin), burglarized(X)$
(S_{10}) $P(burglary(X, yes)|nbrhd(X, bad)) = .4 \leftarrow live\_in(X, wisconsin), burglarized(X)$
(S_{11}) $P(burglary(X, yes)|nbrhd(X, good)) = .15 \leftarrow live\_in(X, wisconsin), \neg burglarized(X)$
(S_{12}) $P(burglary(X, yes)|nbrhd(X, bad)) = .3 \leftarrow live\_in(X, wisconsin), \neg burglarized(X)$
(S_{13}) $P(alarm(X, yes)|tornado(Y, yes)) = .99 \leftarrow live\_in(X, wisconsin), in\_area(X, Y)$
(S_{14}) $P(alarm(X, yes)|tornado(Y, no)) = .1 \leftarrow live\_in(X, wisconsin), in\_area(X, Y)$
(S_{15}) $P(alarm(X, yes)|burglary(X, yes)) = .98$
(S_{16}) $P(alarm(X, yes)|burglary(X, no)) = .05$
(S_{17}) $P(alarm(X, yes)|quake(Y, yes)) = .99 \leftarrow live\_in(X, california), in\_area(X, Y)$
(S_{18}) $P(alarm(X, yes)|quake(Y, no)) = .15 \leftarrow live\_in(X, california), in\_area(X, Y)$
(S_{19}) $P(quake(Y, yes)) = .2 \leftarrow in(Y, california)$
(S_{20}) $P(tornado(Y, yes)) = .1 \leftarrow in(Y, wisconsin)$
(S_{21}) $P(tornado(Y, no)) = .9 \leftarrow in(Y, wisconsin)$
$\cdots\}$

$CB = \{$ $live\_in(X, Z) \leftarrow live\_in(X, Y), in(Y, Z)$
$live\_in(X, Y) \leftarrow in\_area(X, Y)$
$in(madison, wisconsin)\}$

$CR = \{$ Generalized-Noisy-OR$\}$

Fig. 2. Example knowledge base.

nonmonotonic reasoning. Finally, Generalized Noisy-OR is used as the combining rule for all p-predicates.

We can see that a *KB* has a two-layer structure. The lower layer is the normal program *CB* whose semantics is used to select the relevant p-sentences from the upper layer *PB*. That selection process, which is presented in the following sections, is performed dynamically for each given set of context information and set of evidence.

3. Declarative semantics

In this section we give a declarative characterization of the probabilistic information implied by a *KB* in a concrete context with a given set of evidence. The concepts here are defined without reference to any particular query-answering procedure. Following the currently common way of characterizing the semantics of a logic program [9, 8], we define the relevant set of atoms in a bottom-up manner from antecedent to consequent. In contrast, the query-answering procedure presented in Section 4 functions in a top-down manner.

3.1. The relevant knowledge base

It can be difficult for a user to guarantee global consistency of a large probabilisitic knowledge base, especially when we allow context-dependent specifications. We define our semantics so that we need consider only that portion of a knowledge base relevant to a given problem. Thus, if part of the knowledge base is inconsistent, this will not affect reasoning in all contexts.

Definition 4. A set of *evidence* E is simply a set of p-atoms such that $ground(E)$ is coherent. A set of *context information* C is any set of c-atoms.

In a particular inference session, characterized by a set of context information and a set of evidence, only a portion of the *KB* will be relevant. The set of relevant atoms is the set of evidence, the set of atoms whose marginal probability is directly stated, and the set of atoms influenced by these two sets, as indexed by the context information. In constructing the set of relevant p-atoms, we consider only the qualitative dependencies described by probabilistic sentences. If $(P(A_0|A_1,\dots,A_n) = \alpha) \leftarrow L_1,\dots,L_m$ is a ground instance of a sentence in *PB* conforming to domain type constraints and $L_1 \wedge \cdots \wedge L_m$ can be deduced from $completed(C \cup CB)$, then that sentence indicates that A_0 *is directly influenced (or caused) by* $A_1,\dots,A_n$ in the current context. If, in addition, $A_1,\dots,A_n$ are relevant p-atoms then it is natural to consider A_0 as relevant. Let $completed(C \cup CB)$ be the completed logic program with the associated *equality theory* [17] constructed from $C \cup CB$, then we have the following definition of the set of relevant p-atoms.

Definition 5. Assume a set of evidence E, a set of context information C, and a *KB*. The set of *relevant p-atoms* (*RAS*) is defined as the smallest set R statisfying the following conditions:

(i) $ground(E) \subseteq R$;

(ii) if S is a ground instance of a p-sentence in *PB*, conforming to domain constraints, such that $context(S)$ is a logical consequence of $completed(C \cup CB)$ and $ante(S) \subseteq R$ then $cons(S) \in R$;

(iii) if a p-atom A is in R then $Ext(A) \subseteq R$.

The *RAS* is constructed in a way similar to Herbrand least models for Horn programs. Context information is used to eliminate that portion of *PB* not related to the current problem.

Definition 6. Let E be a set of evidence. We define *the set of p-sentences describing the evidence* E, denoted by *EBase*, as $\bigcup_{\{A \in ground(E)\}}(\{P(A) = 1\} \cup \{P(A') = 0 \mid A' \in Ext(A) \wedge A' \neq A\})$.

The relationship between our *KB*s and logic programs is demonstrated in the following definition and property.

Definition 7. Given a set of evidence E, a set of context information C, and a *KB*; the *Horn version of* $\langle E, C, KB\rangle$, denoted by $Horn(E, C, KB)$, is a Horn logic program defined in the following way:

(i) Let R_1 be the set of all $prob(S)$, where S is a ground instance of some p-sentence in *PB*, conforming to domain constraints, such that $context(S)$ is a logical consequence of $completed(C \cup CB)$.

(ii) Let S be the p-sentence $P(A_0|A_1, \dots, A_n) = \alpha$. We define $clause(S)$ to be the set of all Horn clauses $B_0 \leftarrow A_1, \dots, A_n$, where B_0 is an arbitrary element in $Ext(A_0)$.

Then $Horn(E, C, KB) = (\bigcup_{\{S \in R_1\}} clause(S)) \cup (\{Ext(A) \mid A \in ground(E)\})$.

Proposition 8. *Given a set of evidence E, a set of context information C, and a KB; RAS always exists and is equal to the least Herbrand model of Horn(E, C, KB).*

Definition 9. Given a set of evidence E, a set of context information C, and a *KB*; the *set of relevant p-sentences* (*RPB*) is constructed in the following way:

(i) Let R be the set of all $prob(S)$ such that S is a ground instance of a p-sentence in *PB*, $context(S)$ is a logical consequence of $completed(C \cup CB)$, $cons(S) \in RAS$, and $ante(S) \subseteq RAS$.

(ii) If $A \in ground(E)$ and $\neg\exists S \in R : cons(S) \in Ext(A)$ then add to R the following set: $\{P(A) = 1\} \cup \{P(A') = 0 \mid A' \in Ext(A) \wedge A \neq A'\}$.

RPB is the final set R.

If S is a p-sentence in *RPB* then S originates from either (a p-sentence in) *PB* or *EBase*. Condition (ii) augments the knowledge base *KB* with new knowledge about the evidence. The following example illustrates this point.

Example 10. Let $PB = \{P(q(true)|r(true)) = 0.7;\ P(q(false)|r(true)) = 0.3;\ P(q(true)|r(false)) = 0.5;\ P(q(false)|r(false)) = 0.5\}$. Without evidence, *RAS* and *RPB* are both empty. Suppose the set of evidence is $E = \{r(true)\}$. Then the corresponding *RAS* is $\{q(true), q(false), r(true), r(false)\}$ and $RPB = PB \cup \{P(r(true)) = 1, P(r(false)) = 0\}$. Notice that *RPB* contains two p-sentences which are introduced by the second step of Definition 9. These two p-sentences are necessary to build the link matrix for the evidence $r(true)$.

The *RPB* contains the basic influences among p-atoms in *RAS*. In the case of multiple influences represented by multiple sentences, we need combining rules to construct the combined probabilistic influence.

Definition 11. Given a set of evidence E, a set of context information C, and a KB; the *set of combined relevant p-sentences* (*CRPB*) is constructed by applying the corresponding combining rule to each maximal set of sentences $\{S_1, \ldots, S_n\}$ in *RPB* for which the elements have the same consequent.

*CRPB*s are analogous to completed logic programs. We assume that each sentence in *CRPB* describes all random variables that directly influence the random variable in the consequent.

Example 12. Consider our burglary example and suppose we have the evidence $E = \{burglary(john, yes)\}$ and the context $C = \{in\_area(john, madison), burglarized(john)\}$. In this case,
$RAS = \{nbrhd(john, bad), nbrhd(john, average), nbrhd(john, good), burglary(john, yes),$
$burglary(john, no), alarm(john, yes), alarm(john, no), tornado(madison, yes),$
$tornado(madison, no)\}$
$RPB = \{P(nbrhd(john, average)) = 0.4, P(nbrhd(john, bad)) = 0.2,$
$P(nbrhd(john, good)) = 0.4, P(burglary(john, yes)|nbrhd(john, average)) = 0.4,$
$P(burglary(john, yes)|nbrhd(john, good)) = 0.2,$
$P(burglary(john, yes)|nbrhd(john, bad)) = 0.4,$
$P(alarm(john, yes)|tornado(madison, yes)) = 0.99,$
$P(alarm(john, yes)|burglary(john, yes)) = 0.98,$
$P(alarm(john, yes)|burglary(john, no)) = 0.05,$
$P(alarm(john, yes)|tornado(madison, no)) = 0.1,$
$P(burglary(john, no)|nbrhd(john, good)) = 0.8,$
$P(burglary(john, no)|nbrhd(john, average)) = 0.6,$
$P(burglary(john, no)|nbrhd(john, bad)) = 0.6,$
$P(alarm(john, no)|tornado(madison, yes)) = 0.01,$
$P(alarm(john, no)|burglary(john, yes)) = 0.02,$
$P(alarm(john, no)|burglary(john, no)) = 0.95,$
$P(alarm(john, no)|tornado(madison, no)) = 0.9, P(tornado(madison, yes)) = 0.1,$
$P(tornado(madison, no)) = 0.9\}.$
In the *CRPB* the sentences in *RPB* with *alarm*(*john*,.) as the consequent are transformed into sentences specifying the probability of *alarm* conditioned on both *burglary* and *tornado* using a generalization of Noisy-OR:
$P(alarm(john, yes)|tornado(madison, yes), burglary(john, yes)) = 0.9998,$
$P(alarm(john, yes)|tornado(madison, yes), burglary(john, no)) = 0.98,$
$P(alarm(john, yes)|tornado(madison, no), burglary(john, yes)) = 0.99,$
$P(alarm(john, yes)|tornado(madison, no), burglary(john, no)) = 0.9,$
$P(alarm(john, no)|tornado(madison, yes), burglary(john, yes)) = 0.0002,$

$P(alarm(john, no)|tornado(madison, yes), burglary(john, no)) = 0.02,$
$P(alarm(john, no)|tornado(madison, no), burglary(john, yes)) = 0.01, and$
$P(alarm(john, no)|tornado(madison, no), burglary(john, no)) = 0.1,$
The other sentences in *RPB* are carried over to *CRPB* unchanged.

We define a syntactic property of *CRPB* that is necessary in constructing Bayesian networks from the *KB*.

Definition 13. A *CRPB* is *completely quantified* if

(i) for all ground atoms A_0 in *RAS*, there exists at least one sentence in *CRPB* with A_0 in the consequent; and

(ii) for all ground sentences S of the form $P(A_0|A_1,\dots,A_n) = \alpha$ in *CRPB* we have the following property: For all $i = 0,\dots,n$, if A_i is $p(\vec{t}, v)$, where $\vec{t}$ is a list of ground terms, and v' is a value in $VAL(p)$, such that $v \neq v'$, then there exists another ground sentence S' in *CRPB* such that S' can be constructed from S by replacing A_i by $p(\vec{t}, v')$ and α by some α'.

If we think of each ground $obj(A)$, where A is some p-atom, as representing a random variable in a Bayesian network model, then the above condition implies that we can construct a link matrix for each random variable in the model. This is a generalization of constraint (C1) in [12]. We do not require the existence of link matrix for every random variable, but only for the random variables that are relevant to an inference problem.

3.2. Probabilistic independence assumption

In addition to the probabilitistic quantities given in a *PB*, we assume some probabilistic independence relationships specified by the structure of the p-sentences. Probabilistic independence assumptions are used in all related work [29, 4, 11, 24, 12] as the main device to construct a probability distribution from local conditional probabilities. Unlike Poole [24], who assumes independence on the set of consistent "assumable" atoms, we formulate the independence assumption in our framework by using the structure of the sentences in *CRPB*. We find this approach more natural since the structure of the *CRPB* tends to reflect the causal structure of the domain and independencies are naturally thought of causally.

Definition 14. Let R be a set of ground context-free p-sentences and let A and B be two ground p-atoms. We say A is *influenced by* B in R if

(i) there exists a sentence $S \in R$, an atom A' in $Ext(A)$, and an atom B' in $Ext(B)$ such that $A' = cons(S)$ and $B' \in ante(S)$ or

(ii) there exists another ground p-atom C such that A is influenced by C in R and C is influenced by B in R.

Example 15. Continuing the burglary example, *alarm(john,yes)* is influenced by *nbrhd(john,good)*, *nbrhd(john,bad)*, *burglary(john,yes)*, and *tornado(madison,no)*.

Assumption. *We assume that if $P(A_0|A_1,\ldots,A_n) = \alpha$ is in CRPB then for all ground p-atoms $B_1,\ldots,B_m$ which are not in $Ext(A_0)$ and not* influenced by *A_0 in CRPB, A_0 and $B_1,\ldots,B_m$ are probabilistically independent given $A_1,\ldots,A_n$. This means that $P(A_0|B_1,\ldots,B_m,A_1,\ldots,A_n) = P(A_0|A_1,\ldots,A_n)$.*

Example 16. We have *alarm(john, yes)* is probabilistically independent of *nbrhd (john, good)* and *nbrhd(john, bad)* given *burglary(john, yes)* and *tornado(madison, no)*.

3.3. Model theory

We define the semantics by using possible worlds on the Herbrand base. This approach of characterizing the semantics by canonical Herbrand models is widely used in work on logic programming [8, 9]. In our semantics, the context constraints in the context base *CB* and the set of context information *C* act to select the appropriate set of the possible worlds over which to evaluate the probabilistic part of the sentences. In this way they index probability distributions.

The *RAS* contains all relevant atoms for an inference session. We assume that in such a concrete situation, the belief of an agent can be formulated in terms of possible models on *RAS*.

Definition 17. Assume a set of evidence *E*, a set of context information *C*, and a *KB*. A *possible model M* of the corresponding *CRPB* is a set of ground atoms in *RAS* such that for all *A* in *RAS*, $Ext(A) \cap M$ has one and only one element. The atoms in *M* represent the set of atoms assigned the value true; all other atoms are assigned the value false.

A probability distribution on the possible models is realized by probability density assignment to each model. Let *P* be a probability distribution on the possible models, we define $P(A_1,\ldots,A_n)$, where $A_1,\ldots,A_n$ are atoms in *RAS*, as $\sum \{P(w)|w$ is a possible model containing $A_1,\ldots,A_n\}$. We take a sentence of the form $P(A_0|A_1,\ldots,A_n) = \alpha$ as shorthand for $P(A_0,A_1,\ldots,A_n) = \alpha \times P(A_1,\ldots,A_n)$, so that probabilities conditioned on zero are not problematic. We say *P* satisfies a sentence $P(A_0|A_1,\ldots,A_n) = \alpha$ if $P(A_0,A_1,\ldots,A_n) = \alpha \times P(A_1,\ldots,A_n)$ and *P* satisfies *CRPB* if it satisfies every sentence in *CRPB*.

Definition 18. A probability distribution *induced by* the set of evidence *E*, the set of context information *C*, and *KB* is a probability distribution on possible models of *CRPB* satisfying *CRPB* and the independence assumptions implied by *CRPB*.

Example 19. If a *CRPB* is incompletely quantified, many possible distributions may satisfy it. For example, if *CRPB* = $\{P(nbrhd(john, bad)) = 0.5$, $P(tornado(madison, yes)) = 0.6$, $P(tornado(madison, no)) = 0.4\}$ and *nbrhd* can take one of the three values *bad, average, good*, then any probability assignment P^* such that $P^*(nbrhd(john,$

bad)) = 0.5 and $P^*(nbrhd(john, average)) + P^*(nbrhd(john, good)) = 0.5$ is an induced probability distribution.

We define the consistency property only on the relevant part of a *KB*. Since the entire *KB* contains information about various contexts, testing for consistency of such a *KB* may be very difficult.

Definition 20. A completely quantified *CRPB* is *consistent* if

(i) there is no atom in *RAS* which is influenced by itself;

(ii) if $P(A_0|A_1,\dots,A_n) = \alpha$ is in *CRPB* then $P(A_0|A_1,\dots,A_n) = \alpha'$ is not in *CRPB*, for any $\alpha' \neq \alpha$; and

(iii) for all $P(A_0|A_1,\dots,A_n) = \alpha$ in *CRPB*, $\sum\{\alpha_i|P(A'_0|A_1,\dots,A_n) = \alpha_i \in CRPB$ and $obj(A'_0) = obj(A_0)\} = 1$.

Condition (i) rules out cycles and condition (iii) enforces the usual constraint that the probabilities of the values of a random variable sum to one. The following example shows that *PB* may contain cycles but a specific completely quantified *CRPB* is still consistent.

Example 21. Let $PB = \{P(p(true)|p(true)) = 1;\ P(p(true)|p(false)) = 1;\ P(p(false)|p(true)) = 1;\ P(p(false)|p(false)) = 1;\ P(q(true)|r(true)) = 0.7; P(q(false)|r(true)) = 0.3; P(q(true)|r(false)) = 0.5;\ P(q(false)|r(false)) = 0.5\}$. *PB* contains a cycle (p is influenced by itself) and the probability assignment is not reasonable ($P(p(false)|p(true)) = 1$). Assume the set of evidence is $E = \{r(true)\}$. The *CRPB* $\{P(q(true)|r(true)) = 0.7;\ P(q(false)|r(true)) = 0.3;\ P(q(true)|r(false)) = 0.5;\ P(q(false)|r(false)) = 0.5;\ P(r(true)) = 1;\ P(r(false)) = 0\}$ is completely quantified and consistent. Notice that the presence of the evidence E causes the information about q and r to be taken into account and since p is not in *RAS*, the information about it is discarded.

Theorem 22. *Assume a set of evidence E, a set of context information C, and a KB. If RAS is finite and the completely quantified CRPB is consistent, then there exists one and only one induced probability distribution.*

3.4. Probabilistic queries

In one inference session, we can pose queries to ask for the posterior probabilities of some random variables.

Definition 23. A *complete ground query* w.r.t. the set of evidence E and the set of context information C is a query of the form $P(Q) = ?$, where the last attribute of Q is a variable and it is the only variable in Q. The meaning of such a query is: find the posterior probability distribution of $obj(Q)$. If $VAL(Q) = \{v_1,\dots,v_n\}$ then the answer to such a query is a vector $(\alpha_1,\dots,\alpha_n)$, where α_i is the posterior probability that $obj(Q)$ receives the value v_i.

A *complete query* is a query of the form $P(Q) = ?$, where the last attribute of Q is a variable and the other attributes may also contain variables.

Example 24. We can pose the following complete ground query to the example KB: $P(alarm(john, V)) = ?$ to ask for the posterior probability of John's alarm status.

Definition 25. Assume a set of evidence E, a set of context information C, a KB, and a complete ground query $P(Q) = ?$, where $VAL(Q) = \{v_1, \dots, v_n\}$. We say $P(Q) = (\alpha_1, \dots, \alpha_n)$ is a *logical consequence* of $\langle E, C, KB \rangle$ if for all probability distributions P^* induced by E, C, and KB, $\forall i\ (0 \leqslant i \leqslant n) : P^*(\{M | Q_i$ and $ground(E)$ are true in $M\}) = \alpha_i \times P^*(\{M | ground(E)$ are true in $M\})$, where Q_i is Q after replacing $val(Q)$ by v_i.

Example 26. Continuing Example 19, suppose we have the query $P(tornado(madison, V)) = ?$. There are an infinite number of induced probability distributions but in all of them $P(tornado(madison, yes)) = 0.6$ and $P(tornado(madison, no)) = 0.4$. So the two sentences are logical consequences of $CRPB$.

In order to prove that the answers returned by our query answering procedure are correct, we must precisely define the notion of a correct answer to a query.

Definition 27. Assume a set of evidence E, a set of context information C, a KB, and a complete ground query $P(Q) = ?$. Then $P(Q) = (\alpha_1, \dots, \alpha_n)$ is a *correct answer* to the complete ground query $P(Q) = ?$ if $P(Q) = (\alpha_1, \dots, \alpha_n)$ is a logical consequence of $\langle E, C, KB \rangle$.

If $P(Q) = ?$ is a complete query then $P(Q') = (\alpha_1, \dots, \alpha_n)$ is a *correct answer* to the complete query $P(Q) = ?$ if Q' is formed from a ground instance of Q by substituting its last attribute by a new variable name and $P(Q') = (\alpha_1, \dots, \alpha_n)$ is a correct answer to the complete ground query $P(Q') = ?$.

Definition 28. Assume a set of evidence E, a set of context information C, and a KB are given. We say a query answering procedure is *sound* (w.r.t. complete queries) if for any complete query $P(Q) = ?$, any answer returned by the procedure is correct. We say a query answering procedure is *complete* (w.r.t. complete queries) if for any complete query $P(Q) = ?$, any correct answer is returned by the procedure after a finite amount of time.

4. Query answering procedure

Our query answering procedure has the following steps: build the necessary portion of the Bayesian network, each node of which corresponds to an $obj(A)$, where A is a ground p-atom in RAS; update the Bayesian network using the set of evidence E; and output the updated belief of the query nodes. The main idea of the algorithm is to build a *supporting network* for each random variable corresponding to a ground instance of an evidence atom or the query.

4.1. Supporting networks

Definition 29. Let X be a node in a Bayesian network $\mathcal{N}$. The supporting network $\mathcal{M}$ of X in $\mathcal{N}$ is the smallest subnetwork of $\mathcal{N}$ such that

(i) X is in $\mathcal{M}$.

(ii) For any node Y in $\mathcal{M}$, every parent of Y is in $\mathcal{M}$.

(iii) Any link (or link matrix) in $\mathcal{N}$ that relates only to nodes in $\mathcal{M}$ is in $\mathcal{M}$.

Let $X_1, \ldots, X_n$ be nodes in $\mathcal{N}$. The supporting network of $X_1, \ldots, X_n$ is the union of the supporting networks for each X_i, $1 \leqslant i \leqslant n$.

Example 30. The supporting network of the random variable *burglary* in each network shown in Fig. 1 consists of two nodes (*burglary* and *neighborhood*), the link between them, and the corresponding link matrices.

Building a supporting network instead of the whole network to answer a query is justified by the fact that atoms that do not influence either the evidence or the query are irrelevant.

Proposition 31. *Let $\mathcal{N}$ be a Bayesian network, X be a node in $\mathcal{N}$, and E be the evidence that relates to nodes in $\mathcal{N}$. Then the posterior probability of X given the evidence E does not depend on nodes outside the supporting network of X and the nodes related to E.*

The correspondence between a *CRPB* and the network it represents is shown in the following proposition.

Proposition 32. *Assume a set of evidence E, a set of context information C, and a KB are given. If the corresponding CRPB is completely quantified and consistent then:*

(i) *There exists a Bayesian network model $\mathcal{N}$ of the unique probability distribution induced by CRPB.*

(ii) *Let A and B be two ground p-atoms in RAS. B influences A in CRPB if and only if the node corresponding to $obj(B)$ is in the supporting network of the node corresponding to $obj(A)$ in $\mathcal{N}$, and the two nodes are different.*

Each p-sentence in *CRPB* corresponds to a link matrix entry in the Bayesian network. Because of the direct correspondence between *CRPB* and the Bayesian network it represents, we will sometimes refer to "the portion of *CRPB* supporting A (or $obj(A)$)", where A is a ground p-atom, to indicate the supporting network of $obj(A)$. Similarly, we will refer to "the portion of *CRPB* supporting a set of ground p-atoms."

4.2. Q-procedure

In this section we present an algorithm, Q-procedure, for answering complete queries $P(Q) = ?$, given E, C, and KB. We assume that the *domains are finite*. Because of the similarity between *RPB* and the Horn logic program *Horn*(E, C, KB) (see Section 3.1), Q-procedure's performance is similar to that of SLD resolution with the depth-first search strategy and with the formation of lemmas during its execution. Every time all p-sentences in *RPB* which have the same p-atom B as consequent have been collected, Q-procedure stores them in a node of a partially generated Bayesian network and later uses B as a lemma to avoid building the proof tree of B again. Q-procedure works correctly on acyclic *PB*s.

Definition 33. A *PB* is called *acyclic* if there is a mapping $P\_ord()$ from the set of ground instances of p-atoms into the set of natural numbers such that (1) For any ground instance $(P(A_0 \leftarrow A_1, \ldots, A_n) = \alpha) \leftarrow L_1, \ldots, L_m$ of some clause in *PB*, $P\_ord(A_0) > P\_ord(A_i), \forall i : 1 \leqslant i \leqslant n$. (2) For any ground p-atom A, $P\_ord(A) = P\_ord(A'), \forall A' \in Ext(A)$.

For example, our burglary problem has an acyclic *PB*. The expressiveness of acyclic logic programs is demonstrated in [1]. We hope that acyclic *PB*s have equal importance. To the best of our knowledge, *PB*s with cycles are considered problematic and all *PB*s considered in the literature are acyclic, e.g. [24].

The pseudo-code for Q-procedure is shown in Fig. 3. The FOR loop constructs the supporting network for the set of evidence and the final BUILD-NET call augments the constructed network with the supporting network of ground instances of the query. UPDATE(NET, E) is any probability propagation algorithm on Bayesian networks.

The supporting networks are constructed via calls to the BUILD-NET function. BUILD-NET receives as input an atom, whose supporting network needs to be ex-

```
Q-PROCEDURE
BEGIN
        NET:= {};
        $EBase := \bigcup_{\{A \in ground(E)\}}(\{P(A) = 1\} \cup \{P(B) = 0 | B \in Ext(A) \wedge A \neq B\})$;
        $Extended\_PB := PB \cup EBase$;
        { Build the subnetwork that supports the evidence }
        FOR each $E_i \in E$ DO BEGIN
                temp := BUILD-NET($E_i$, NET, $Extended\_PB$);
        END;
        { Build the subnetwork that supports the ground instances of the query }
        SUBSS := BUILD-NET ($Q$, NET, $Extended\_PB$);
        { Prune the isolated portions of the network }
        FOR each $obj(A)$ such that $A \in ground(E)$ DO
                IF there is no path connecting $obj(A)$
                   and a node corresponding to a ground instance of the query
                THEN delete the supporting network of $obj(A)$
        { Perform the probability propagation on NET}
        UPDATE(NET,$E$);
        { Output posterior probabilities }
        FOR each $\theta$ in SUBSS output the probability values at node $obj(Q\theta)$;
END.
```

Fig. 3. Query processing procedure.

plored. It updates the NET, which might have been partially built. The return value of the function is a set of ground substitutions such that the resulting NET contains the supporting network for the ground instances of the input atom corresponding to those substitutions. BUILD-NET frequently calls SLDNF to answer queries on $CB \cup C$.

In BUILD-NET (Fig. 4), the *RS* variable is used to collect all sentences in *RPB* that have the currently considered atom as consequent. The COMBINE function uses the appropriate combining rule in CR and the sentences in RS to generate the corresponding sentences in the *CRPB*. The variable SUBSS stores all returned substitutions. The recursive structure of BUILD-NET is realized through the function BUILD-C-NET which builds the supporting network for the atoms in the antecedents of relevant sentences.

BUILD-C-NET (Fig. 5) receives as input a conjunction of atoms, whose supporting networks need to be explored. It updates the NET, which might have been partially built. The return value of the function is a set of substitutions such that for each substitution all the ground instances resulting from applying the substitution to the atoms in the conjuction appear in the resulting net.

Example 34. To illustrate the functioning of Q-procedure, we trace through it for a particular query. We show nested function calls with indentation. Consider the burglary example and suppose we want to know the posterior probability that John has been burglarized given that he lives in Madison and that his alarm has sounded. Then we can take $C = \{in\_area(john, madison)\}$, $E = \{alarm(john, yes)\}$, and the query as $P(burglary(john, X)) = ?$.

First, the following sentences are inserted into *PB* to obtain *Extended_PB*: $P(alarm(john, yes)) = 1$ and $P(alarm(john, no)) = 0$. Next BUILD-NET is called: BUILD-NET($alarm(john, yes), NET, Extended\_PB$). In BUILD-NET, the parameter A contains $alarm(john, yes)$ and NET is empty. There are two p-atoms in $Ext(A)$: $alarm(john, yes)$ and $alarm(john, no)$. Due to space limitations, we show only a partial trace for the first one.

$B = alarm(john, yes)$.

(1) $S = S_{13}$, $\theta^S = \{X|john\}$, $\theta^C = \{Y|madison\}$.

Because $ante(S) \neq \{\}$, BUILD-C-NET($tornado(madison, yes), NET, Extended\_PB$) is invoked.

It simply calls BUILD-NET with $tornado(madison, yes)$.

In BUILD-NET: $A\_set = \{tornado(madison, yes)\}$.

(1) $B = tornado(madison, yes), S = S_{20}$, $\theta^S = \{Y|madison\}$, $\theta^C = \{\}$.

Because $ante(S) = \{\}$, $RS = \{P(tornado(madison, yes) = 0.1\}$, $CRS = RS$.

The node $obj(tornado(madison, yes))$ is created and the link matrix entry $P(tornado(madison, yes)) = 0.1$ is inserted.

(2) $B = tornado(madison, no), S = S_{21}$, $\theta^S = \{Y|madison\}$, $\theta^C = \{\}$.

Because $ante(S) = \{\}$, $RS = \{P(tornado(madison, no) = 0.9\}$, $CRS = RS$.

The entry $P(tornado(madison, no)) = 0.9$ is inserted into the node $obj(tornado(madison, yes))$.

```
BUILD-NET
Function BUILD-NET( A: a p-atom; var NET: a Bayesian net; $Extended\_PB$:
     a set of probabilistic sentences): set of substitutions;
BEGIN
  $A\_set := ground(A)$; SUBSS := {};
  FOR each $A\theta \in A\_set$ DO BEGIN
    FOR each $B \in Ext(A\theta)$ DO BEGIN
     IF there is a node in NET corresponding to $obj(A\theta)$
        and the link matrix entry for $val(B)$ exists
     THEN  SUBSS := SUBSS $\cup \{\theta\}$
     ELSE
        BEGIN
             RS := {};
             FOR each $S \in Extended\_PB$ such that there exists
                  an mgu $\theta^S$ of $B$ and $cons(S)$ DO
                  { Assume that if context(S)={} then $\theta^C = \{\}$}
                  FOReach $\theta^C$ which is a computed answer from
                       $(\leftarrow context(S)\theta^S, C \cup CB)$ returned by SLDNF DO
                       IF $ante(S) = \{\}$
(S.1)                  THEN RS := RS $\cup\{prob(S)\theta^S\theta^C\}$
                       ELSE  BEGIN
                                 SUBSS1:= BUILD-C-NET(
                                      $ante(S)\theta^S\theta^C, NET, Extended\_PB$);
(S.1')                            RS:= RS $\cup\{prob(S)\theta^S\theta^C\theta' | \theta' \in SUBSS1\}$
                             END;
(S.2)        IF $RS$ contains an p-sentence that originated from $PB$ THEN
                  Delete from $RS$ all p-sentences that originated from $EBase$;
             $CRS := COMBINE(RS, CR)$;
             (//1//)
             IF     CRS is not empty THEN BEGIN
                    SUBSS := SUBSS $\cup \{\theta\}$;
                    IF $obj(A\theta)$ is not in NET THEN
                       Construct node $obj(A\theta)$ and Build links from the parents;
                    Assign the corresponding link matrix entries;
             END
        (//2//)
        END;
    END;
  END;
  (//3//)
  RETURN SUBSS;
END;
```

Fig. 4. BUILD-Net function

Now NET contains the one node $obj(tornado(madison, .)$ with its link matrix, BUILD-C-NET returns and $RS = \{P(alarm(john, yes)|tornado(madison, yes)) = 0.99\}$.

(2) $S = S_{14}$, $\theta^S = \{X|john\}$, $\theta^C = \{Y|madison\}$.

Because $ante(S) \neq \{\}$, BUILD-C-NET($tornado(madison, no), NET, Extended\_PB$) is invoked.

It simply calls BUILD-NET with $tornado(madison, no)$.

```
BUILD-C-NET
FunctionBUILD-C-NET($A_1 \wedge \ldots \wedge A_n$: a conjunction of p-atoms; var NET:
        a Bayesian net;$Extended\_PB$:a set of p-sentences):set of substitutions;
var SUBSS, SUBSS1: set of substitutions;
BEGIN
        SUBSS := BUILD-NET ($A_1$, NET, $Extended\_PB$);
        IF $n > 1$ THEN BEGIN
            SUBSS1:= SUBSS;
            SUBSS:={};
            FOR each $\theta \in$ SUBSS1 DO
                SUBSS:= SUBSS$\cup\{\theta\theta' \mid \theta' \in$ BUILD-C-NET(
                        $(A_2 \wedge \ldots \wedge A_n)\theta, NET, Extended\_PB)\}$;
        END;
        RETURN SUBSS;
END.
```

Fig. 5. BUILD-C-NET function.

In BUILD-NET: because the node $obj(tornado(madison, no))$ and its link matrix entries exist, BUILD-NET just returns $\{\}$.

Now $RS = \{P(alarm(john, yes)|tornado(madison, yes)) = 0.99,$
$P(alarm(john, yes)|tornado(madison, no)) = 0.1\}$.

(3) $S = S_{15}$, $\theta^S = \{X|john\}$, $\theta^C = \{\}$.

Because $ante(S) \neq \{\}$, BUILD-C-NET$(burglary(john, yes), NET, Extended_PB)$ is invoked.

It simply calls BUILD-NET with $burglary(john, yes)$.

In BUILD-NET: $A\_set = \{burglary(john, yes)\}$.

(1) $B = burglary(john, yes), S = S_6$, $\theta^S = \{X|john\}$, $\theta^C = \{\}$.

Since $ante(S) \neq \{\}$, BUILD-C-NET$(nbrhd(john, average), NET, Extended_PB)$ is invoked.

It simply calls BUILD-NET with $nbrhd(john, average)$.

...

A process similar to the construction of node $obj(tornado(madison, .))$ occurs and the resulting NET contains two nodes with the associated link matrices.

(2) $B = burglary(john, yes), S = S_{11} \cdots$

In this case, because NET already contains the node $obj(nbrhd(john, .))$, BUILD-NET simply returns $\{\}$ without exploring subgoals.

...

(3) $B = burglary(john, yes), S = S_{13} \cdots$

...

$RS = \{P(burglary(john, yes)|nbrhd(john, average)) = 0.4,$
$P(burglary(john, yes)|nbrhd(john, good)) = 0.15,$
$P(burglary(john, yes)|nbrhd(john, bad)) = 0.3\}$

$CRS = RS$ and the node $obj(burglary(john, .))$ is created with the entries for $burglary(john, yes)$.

(4) Now $B = burglary(john, no)$ and a process similar to the above will create the link matrix entries for $burglary(john, no)$.

...

(4) $S = S_{16}$, $\theta^S = \{X|john\}$, $\theta^C = \{\}, \cdots$

...

The final RS for $alarm(john, yes)$ is

$RS = \{P(alarm(john, yes)|tornado(madison, yes)) = 0.99,$
$P(alarm(john, yes)|burglary(john, yes)) = 0.98,$
$P(alarm(john, yes)|burglary(john, no)) = 0.05,$
$P(alarm(john, yes)|tornado(madison, no)) = 0.1\}.$

The corresponding combining rule is used to generated the corresponding set of p-sentences in *CRPB* in a way similar to what is shown in Example 12. The node $obj(alarm(john, .))$ is created and the link matrix entries for $alarm(john, yes)$ are inserted.

After the first *BUILD-NET* call in the main procedure, the network in Fig. 1(a) is created. The *BUILD-NET* call with the query as input simply returns the substitution $\{\}$ without spending time to build a supporting network. NET is passed to a standard probability updating procedure for evaluation.

The complexity of Q-procedure is comparable to that of SLDNF. The performance of the function BUILD-NET is similar to that of the SLD-proof procedure. BUILD-NET invokes SLDNF many times during its execution. BUILD-NET gains efficiency by maintaining lemmas in the contructed NET. The complexity of Q-procedure depends on the size of the constructed network and the context base. Because updating probabilities on a Bayesian network is an NP-hard problem [5], the main bottleneck lies in the UPDATE procedure.

4.3. Correctness of Q-procedure

In Q-procedure, we construct a portion of the entire network containing the supporting network of the evidence and the query.

Proposition 35. *Assume a complete query $P(Q) = ?$, a set of evidence E, a set of context information C, and a KB. If* (1) *the domains are finite,* (2) *CRPB is completely quantified and consistent,* (3) *the proof procedure for $C \cup CB$ is sound and complete w.r.t. any context query generated by Q-procedure, and* (4) *PB is acyclic; then:*

(i) *If $A \in ground(Q)$ then $obj(A)$ is a node in the constructed network if and only if $A \in RAS$.*

(ii) *The network constructed by Q-procedure* (*before the pruning step in the main procedure*) *contains the supporting network of $\{obj(A) | A \in (ground(Q) \cap RAS) \cup ground(E)\}$.*

Theorem 36 (Soundness and completeness). *Assume a set of evidence E, a set of context information C, and a KB are given. If* (1) *the domains are finite,* (2) *CRPB is* *completely quantified and consistent,* (3) *the proof procedure for* $C \cup CB$ *always stops after a finite amount of time and is sound and complete w.r.t. any context query generated by Q-procedure, and* (4) *PB is acyclic; then Q-procedure is sound and complete w.r.t. complete queries.*

4.4. Completeness of Q-procedure for acyclic knowledge bases

In this section, we show that Q-procedure is complete for a large class of *KB*s, which is characterized by acyclicity.

Definition 37. A *CB* is called acyclic if there is a mapping $C\_ord()$ from the set of ground instances of c-atoms into the set of natural numbers such that for any ground instance $A \leftarrow L_1, \ldots, L_n$ of some clause in *CB*, $C\_ord(A) > C\_ord(L_i)$, $\forall i : 1 \leqslant i \leqslant n$, where we extend $C\_ord()$ by $C\_ord(\neg A) = C\_ord(A)$, for any c-atom A.

A *KB* is acyclic if both *PB* and *CB* are acyclic.

Theorem 38. *Assume a set of evidence E, a set of context information C, and a KB are given. If* (1) *the domains are finite,* (2) *KB is acyclic, and* (3) *CRPB is completely quantified and consistent, then there is a version of the SLDNF proof procedure such that Q-procedure is sound and complete w.r.t. complete queries.*

5. Implementation

We have implemented a version of Q-procedure called BNG (Bayesian Network Generator)[2]. The BNG system is written in CommonLisp and interfaces to the IDEAL [27] Bayesian network inference system. BNG differs from Q-procedure in three primary respects. First, it does not reason with a context base. Contextual reasoning is limited to matching the contexts of rules against a set of ground atomic formulas specifying the present context. Negation is implemented via negation as failure. Second, BNG is capable of resoning with temporally indexed random variables, while the framework in this paper cannot deal with time. Third, the BNG system performs d-sepration based pruning of the generated network to further eliminate irrelevant nodes.

The algorithm proceeds by first generating the network and then pruning d-separated nodes. By simply backward chaining on the query and on the evidence atoms the generation phase generates all relevant nodes and avoids generating barren nodes, which are nodes below the query that have no evidence nodes below them. Such nodes are irrelevant to the computation of $P(Q|E)$ [25]. The pruning phase involves traversing the generated network using a modified depth-first search originating at the query. Only

[2] The code is available at http://www.cs.uwm.edu/faculty/haddawy.

those nodes reachable via an active path from the query are visited and marked as reachable. Upon termination of the search, only reachable nodes are retained.

6. Applications

6.1. Plan projection

The usefulness of the contextual reasoning method presented in this paper depends on the presence of domains in which deterministic information can be identified and used as context to index probabilistic relations. Probabilistic plan projection is one such domain. Probabilistic planning problems are typically described in terms of both causal rules describing the effects of actions and persistence rules describing the behavior of random variables in the absence of actions. If we assume that at most one action is performed at any one time, then only one causal or persistence rule will be applicable over a time period. Since when evaluating a plan, the performance of an agent's own actions is deterministic knowledge – the agent knows whether or not it plans to attempt an action –, actions can be used as context to index the causal and persistence rules. In related papers [22, 13] we present a temporal variant of our logic and demonstrate the effectiveness of applying the BNG system to reasoning about the effects of interventions and medications on the state of patients in cardiac arrest. The knowledge base contains one rule for each medication and intervention, describing its effect on the patient's heart rhythm, as well as one rule describing how the heart rhythm changes in the absence of any action. In this domain the use of context constraints results in large reductions in the sizes of generated networks and the link matrices contained in them.

6.2. Taxonomic inheritance

Lin and Goebel [16] pointed out the need to integrate taxonomic and probabilistic reasoning for diagnostic problem solving. By combining logic programs and probabilistic causal relationships, our framework can be used for that purpose. The concept of context is also used in Ng and Subramanian's Empirical Probabilistic Logic Programs [20, 18]. The following is a reformulation in our framework of an example which was used in [20] to demonstrate the need for context.

Example 39. We represent the fact "normally elephants are gray, but royal elephants normally are white" by the following p-sentences in *PB*:
$PB = \{Pr(color(X, white)) = 0.9 \leftarrow royal\_elephant(X);$
$Pr(color(X, gray)) = 0.1 \leftarrow royal\_elephant(X);$
$Pr(color(X, white)) = 0.01 \leftarrow elephant(X), \neg ab(X);$
$Pr(color(X, gray)) = 0.99 \leftarrow elephant(X), \neg ab(X)\}.$

The *ab* predicate is the well-known means of representing abnormality in default reasoning. In *CB*, we can utilize the logic programming techniques for representing default knowledge to represent the fact "*royal elephants are abnormal elephants*":

$CB = \{elephant(X) \leftarrow royal\_elephant(X);$
$elephant(X) \leftarrow normal\_elephant(X); ab(X) \leftarrow royal\_elephant(X)\}$.

If we know that *Alex* is a royal elephant then $CB \cup \{royal\_elephant(alex)\}$ can be used to select the correct probabilistic information about the color of *Alex*.

7. Related work

7.1. Knowledge-based model construction

This work extends the earlier work of one of the authors [12] in several significant ways. That work did not address the problem of representing context-sensitive knowledge. In addition, it imposed a number of constraints (C1–C4) on the language in order to guarantee that a set of ground instances of the rules in the knowledge base was isomorphic to a Bayesian network. In this work we drop these constraints, which increases the expressive power of the language but requires us to specify combining rules. Our probabilistic independence assumption is simpler than that of d-separation for a knowledge base used in [12].

Our representation language has many similarities to Breese's Alterid [4]. Alterid can represent the class of Bayesian networks and influence diagrams with discrete-valued nodes. Influence diagrams are similar to Bayesian networks but include decision and value nodes. Breese also uses a form of predicates similar to our p-predicates. Alterid also has probabilistic rules with context constraints and uses logic program clauses to express relations among the context atoms. But in contrast to our framework, Breese mixes logic program clauses with probabilistic sentences by using the logic program clauses to infer information about evidence. In contrast, we separate logic program clauses from probabilistic sentences. In our framework, pure logical dependencies between p-atoms are represented by 0 and 1 probability values, which can be less efficient than representing them with logical sentences. Breese does not provide a semantics for the knowledge base. As a result, his paper does not prove the correctness of his query answering procedure. Breese's procedure does both backward and forward chaining. Because Q-procedure only chains backwards, we can extend the procedure for some infinite domains. In Breese's framework, a set of ground instances of rules may not represent a single network. In this case, the construction algorithm must choose between networks to generate.

Goldman and Charniak [10, 11] present a language called Frail3, for representing belief networks, and outline their associated network construction algorithm. In their framework, rules, evidence, and the emerging network are all stored in a common data base. They also allow additional information to be stored in a separate knowledge base. For example, for the natural language processing applications they consider, the knowledge base can be used to store a dictionary of word meanings. The knowledge base plays a role similar to that of our context base.

Frail3 represents network dependencies by rules with variables. The rules in Frail3 differ from ours in that they contain embedded control information. Frail3 uses two operators to express network construction rules: $\rightarrow$ and $\rightarrow\leftarrow$. The $\rightarrow$ rules have the form ($\rightarrow$ *trigger consequent*:*prob pforms*). When a statement unifying with trigger is added to the database, the bindings are applied to the consequent and the resulting statement is added to the database, along with a possible probabilistic link. The atoms within the rule with which the link is associated and the direction of the link are specified independent of the direction of rule application. The $\rightarrow\leftarrow$ operator is used to incorporate information from a knowledge base and to perform network expansions conditional on the previous network contents. Its first function is similar to our notion of context constraints. The language includes numerous combining rules, such as noisy-OR. Networks are generated by a forward and backward chaining TMS type system. A rigorous semantics for the Frail3 representation language is not provided.

Poole [24] expresses an intention similar to ours: "there has not been a mapping between logical specifications of knowledge and Bayesian network representations ...". He provides such a mapping using probabilistic Horn abduction theory, in which knowledge is represented by Horn clauses and the independence assumption of Bayesian networks is explicitly stated. His work is developed along a different track than ours, however, by concentrating on using the theory for abduction. Our approach has several advantages over Poole's. We do not impose as many constraints on our representation language as he does. Probabilistic dependencies are simply directly represented in our language, while in Poole's language they are indirectly specified through the use of special predicates in the rules. Our probabilistic independence assumption is more intuitively appealing since it reflects the causality of the domain. Our framework is representationally richer than Poole's by allowing context-dependent specification.

Bacchus [2] shows how to translate between a subclass of Bayesian networks and a subclass of his first-order statistical probability logic. He does not provide a network construction procedure.

7.2. *Logic programming*

In the quest to extend the framework of logic programming to represent and reason with uncertain knowledge, there have been several attempts to add numeric representations of uncertainty to logic programming languages [28, 7, 3, 15, 19, 21, 18]. Of these attempts, the only one to use probability is the work of Ng and Subrahmanian [21]. In their framework, a probabilistic logic program is an annotated Horn program. A typical example clause in a probabilistic logic program, taken from [21], is $path(X,Y)$: $[0.85, 0.95] \leftarrow a(X,Y)$: $[1,1]$, which says that if the probability that a type A connection is used lies in the interval [1,1] then the reliability of the path is between 0.85 and 0.95. As this example illustrates, their framework does not conveniently employ conditional probability, which is the most common way to quantify degrees of influence in probabilistic reasoning [23] and in probabilistic expert systems. In [19], the authors allow clauses to be interpreted as conditional probability statements, but they do not employ the causality concept and the independence assumption to gain inference efficiency. Their proof procedure simply constructs and solves a system of linear equations which can be enormous in size.

8. Conclusions and future research

This paper has presented a method for answering queries from context-sensitive probabilistic knowledge bases. Context is used to facilitate specification of probabilistic knowledge and to reduce the complexity of inference by focusing attention on only the relevant portions of that knowledge. A formal framework has been provided in which the correctness and completeness of Bayesian network construction procedures can be formulated and proven.

The knowledge bases in the current framework have two fixed layers: a probabilistic base above a deterministic base. This architecture is used to support our concept of context and to make reasoning more efficient. It implies that if we have a deterministic relationship between random variables, that relationship must be represented with conditional probabilities of 0 and 1. We are working on an extension of the framework that allows a more flexible mixing of probabilistic and deterministic reasoning. The extended framework would allow for reasoning with deterministic relationships between random variables and for reasoning about both probabilistic and deterministic context. We also plan to extend the framework to include infinite domains and continuous random variables. The current framework does not have the concepts of decision and utility. In future work, we plan to investigate the representation of decision models and efficient procedures for finding optimal or good decision strategies.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments.

Appendix

Proof of Proposition 8. We prove that for every set R of ground p-atoms, R satisfies the conditions (i), (ii), and (iii) of Definition 5 only if R is a (Herbrand) model of $Horn(E,C,KB)$. If R satisfies these conditions and is not a model of $Horn(E,C,KB)$ then there exists a ground instance $B \leftarrow A_1,\ldots,A_n$ of a clause in $Horn(E,C,KB)$ such that $A_1,\ldots,A_n \in R$ but $B \notin R$. That clause cannot be a ground instance of a clause in *EBase*. Hence, there exists a p-sentence $P(A|A_1,\ldots,A_n) = \alpha$ in R_1 such that $A \in Ext(B)$. Because R satisfies conditions (ii) and (iii), B must be in R. We have a contradiction.

The least Herbrand model M of $Horn(E,C,KB)$ exists [17]. It is easy to see that M satisfies conditions (i) and (ii) of Definition 5. It violates condition (iii) only if there is a p-atom $A \in M$ such that $Ext(A) \not\subseteq M$. Because $A \in M$ and obviously A cannot be a ground instance of an evidence, there must exist $A \leftarrow B_1,\ldots,B_n$ in $Horn(E,C,KB)$ such that $B_1,\ldots,B_n \in M$ [17]. By the definition of $Horn(E,C,KB)$, $Ext(A)$ must be a subset of M, which is a contradiction.

Hence, the smallest set satisfying the conditions (i), (ii), and (iii) is the least model of $Horn(E,C,KB)$. □

Proof of Theorem 22. Let M be a possible model of *CRPB*. We arrange the atoms in M in the order of the influenced-by relationship: If A_i is influenced by A_j in *CRPB* then $j > i$. Let P^* be a probability distribution induced by *CRPB*. We have $P^*(M) = P^*(\bigwedge_{i=1} A_i) = P^*(A_1 \mid \bigwedge_{i=2} A_i) \times P^*(\bigwedge_{i=2} A_i) = P(A_1 \mid A_{11},\ldots,A_{1n}) \times P^*(\bigwedge_{i=2} A_i) = \alpha \times P^*(\bigwedge_{i=2} A_i)$, where $P(A_1 \mid A_{11},\ldots,A_{1n}) = \alpha$ is a sentence in *CRPB* with $\{A_{11},\ldots,A_{1n}\} \subseteq M$, by the independence assumption. By using induction on the length of the sequence, we get a unique P^*. □

Proof of Proposition 31. The posterior probability of X given E is determined from the prior probabilities $P(X,E)$ and $P(E)$. We can arrange nodes to $\mathcal{N}$ in the order $X_1,\ldots,X_n$ such that no node preceds its parents. Let $X_i = v_i$ be the assertion that the node X_i takes the value v_i. Assume $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n$ and we want to evaluate the prior probability $P(\bigwedge_{j=1}^m (X_{i_j} = v_{i_j}))$. We have $P(\bigwedge_{j=1}^m (X_{i_j} = v_{i_j})) = \sum_{\pi_{i_m}} P(\bigwedge_{j=1}^m (X_{i_j} = v_{i_j}) \wedge \pi_{i_m}) = \sum_{\pi_{i_m}} (P(X_{i_m} = v_{i_m} \mid \pi_{i_m}) \times P(\bigwedge_{j=1}^{m-1} (X_{i_j} = v_{i_j}) \wedge \pi_{i_m}))$, where π_{i_m} is an arbitrary value assignment to the parents of X_{i_m}. By induction on m and i_m, we can easily show that $P(\bigwedge_{j=1}^m (X_{i_j} = v_{i_j}))$ only depends on the supporting network of X_{i_j}, $j = 1,\ldots,m$. □

Proof of Proposition 32. We prove them in sequence.

(i) Simply apply Corollary 4, Ch. 3 of [23]. Here each node of the network corresponds to an $obj(A)$, A is an atom in *RAS*; and there is a link from node $obj(B)$ to node $obj(A)$ if there is a probabilistic sentence $P(A \mid \ldots,B,\ldots) = \alpha$ in *CRPB*.

(ii) Follows directly from the definitions of supporting networks and influence relationship. □

The following lemma shows the properties and invariants at various points in BUILD-NET procedure.

Lemma 40 (BUILD-NET Fundamental Lemma). *Assume a set of evidence E, a set of c-atoms C, and a KB. Furthermore, assume* (1) *CRPB is completely quantified*; (2) *SLDNF is sound and complete w.r.t. any query passed by BUILD-NET*; *and* (3) *PB is acyclic.*

Then

(P-ACY) On any sequence of recursive calls of BUILD-NET, no ground p-atom $A\theta$ *appears more than once.*

(P-BN) Let A be the input p-atom of BUILD-NET. The followings are invariants at various points in BUILD-NET:

(1.1) *At point* (//**1**//):

(1.1.1) $RS = \{S \mid S \in RPB \wedge cons(S) = B\}$.

(1.1.2) $\forall S \in RS, \forall A \in ante(S) : obj(A) \in NET$.

(1.1.3) *RS contains only ground p-sentences.*

(1.1.4) $\forall S \in CRS, \forall A \in ante(S) : obj(A) \in NET$.

(1.2) *At point* (//**2**//):

(1.2.1) *NET contains the supporting networks of nodes in NET, except that:* (1) *the link matrix of the node $obj(A\theta)$ generated by the current BUILD-NET execution lacks the entries corresponding to all $B \in Ext(A\theta)$ which have not been processed by BUILD-NET so far;* (2) *the link matrices of the node(s) generated by unfinished BUILD-NET calls (at higher levels) lack the entries corresponding to all B which have not been fully processed by the corresponding BUILD-NET so far.*

(1.3) *At point* (//**3**//):

(1.3.1) *NET contains the supporting networks of nodes in NET, except that the link matrices of the node(s) generated by unfinished BUILD-NET calls (at higher levels) lack the entries corresponding to all B which have not been fully processed by the corresponding BUILD-NET so far.*

(1.3.2) *If this BUILD-NET execution was not invoked by BUILD-C-NET (that means it was invoked by the main procedure) then NET contains the supporting network of any node in NET.*

(1.3.3) $\theta \in SUBSS$ *if and only if* $\exists S \in CRPB : A\theta = cons(S)$ *if and only if* $A\theta \in RAS$ *if and only if* $obj(A\theta) \in NET$.

(P-BCN) Let $A_1 \wedge \cdots \wedge A_n$ be the input conjunction of BUILD-C-NET. Then

(2.1) *If θ is one of the substitutions returned by BUILD-C-NET, then θ is a ground substitution for each A_i and $A_i\theta \in RAS$, $i = 1,\ldots,n$.*

(2.2) *If θ is a ground substitution for all A_i and $A_i\theta \in RAS$, $i = 1,\ldots,n$ then there is a substitution θ' returned by BUILD-C-NET such that $A_i\theta = A_i\theta'$, $i = 1,\ldots,n$.*

In order to prove Lemma 40, we need the following property.

Lemma 41. *Assume a set of evidence E, a set of c-atoms C and a KB are given. Let A be a ground p-atom.*

(1) $\exists S \in RPB : cons(S) = A$ *if and only if* $\exists S \in CRPB : cons(S) = A$.

(2) *If CRPB is completely quantified then* $A \in RAS$ *if and only if* $\exists S \in RPB : cons(S) = A$.

Proof of Lemma 41. (1) Follows from the definition of combining rules: the input of a combining rule is a set of p-sentences in *RPB* and its output is not empty if the input set is not empty.

(2) The IF direction is obvious. We prove the ONLY IF part. If $\exists S \in RPB$: $cons(S) \in Ext(A)$ then $\exists S_1 \in CRPB : cons(S_1) = cons(S) \in Ext(A)$. Because *CRPB*

is completely quantified, $\exists S_2 \in CRPB : cons(S_2) = A$ and $\exists S_3 \in RPB : cons(S_3) = A$. Hence, if $\neg\exists S \in RPB : cons(S) = A$ then $\neg\exists S \in RPB : cons(S) \in Ext(A)$. Consider $RAS - Ext(A)$, it satisfies the requirements on RAS and smaller than RAS. A contradiction. □

Proof of Lemma 40. (P-ACY) Follows from the facts that (1) in each execution of BUILD-NET the input atom is instantiated to ground and (2) the function call to BUILD-C-NET transfers only atoms with smaller *P_ord*'s.

(P-BN and P-BCN) Because of the condition (c) of combining rules, (1.1.4) always follows directly from (1.1.2).

In each BUILD-NET invocation, if a ground instance $A\theta$ is explored, all p-atoms in $Ext(A\theta)$ are also explored. Because *CRPB* is completely quantified, by Lemma 41, if there are p-sentences in *RPB* whose consequents are $A\theta$ and $B \in Ext(A\theta)$ then there are also p-sentences in *RPB* whose consequents are B. Hence, we can deduce (1.3.1) using (1.1.1) and (1.2.1).

(1.2.1) NET starts out with an empty set, which satisfies both (1.2.1) and (1.3.1). There is at most one change to NET which occurs between the two consecutive times the execution reaches either point (//**2**//) or (//**3**//) or between the first invocation of BUILD-NET and the first time (//**2**//) or (//**3**//) is reached. That possible change is: if *CRS* is not empty then there may be a new node created for $obj(A\theta)$ and the link matrix entries corresponding to B are inserted. For other nodes in *NET*, the result follows from either property (1.3.1) which holds right after an exit from a previous BUILD-NET call or when NET is empty or property (1.2.1) which holds at the end of the previous loop.

(1.3.2) follows directly from (1.3.1).

In (1.3.3), "$\exists S \in CRPB : A\theta = cons(S)$ if and only if $A\theta \in RAS$" follows from Lemma 41. To prove "$\theta \in SUBSS$ if and only if $\exists S \in CRPB : A\theta = cons(S)$", notice that θ can only be added into *SUBSS* if $obj(A\theta)$ is already in NET or *CRS* and *RS* are not empty. In the first case, by (P-ACY) $obj(A\theta)$ cannot be being processed by a higher level BUILD-NET call. We use (1.3.1) to get the result. In the second case, the result follows from (1.1.1). By dividing into the same two cases, it is also easy to see that "$\theta \in SUBSS$ if and only if $obj(A\theta) \in NET$" is true.

To prove (1.1.3), we observe that the returned substitutions of BUILD-NET are ground substitutions. The consequent of an arbitrary p-sentence in *RS* is equal to B, which is ground. The substitutions returned by BUILD-C-NET are combinations of ground substitutions returned by BUILD-NET. Hence, the ancecedent of any p-sentence in *RS* is a conjunction of ground p-atoms.

We prove the rest of the lemma item by item and by induction on the largest depth k of the sequences of recursive calls on BUILD-NET:

(P-BN) *Base case* $k = 0$: In this case BUILD-NET makes no recursive call to itself. That means there is no call to BUILD-C-NET.

(1.1.1) We prove $RS \supseteq \{S \mid S \in RPB \wedge cons(S) = B\}$: Assume $S_0 \in RPB$ and $cons(S_0) = B$. We will show $S_0 \in RS$. There are two cases.

(1) In the first case S_0 is introduced by the second part of Definition 9. It is easy to see that S_0 is inserted into *RS* by the statement (S.1) of BUILD-NET because *EBase* $\subseteq$ *Extended_PB*. We show that S_0 is not deleted from *RS* by statement (S.2). Because S_0 is introduced by the second step of the definition of *RPB*, there is no p-sentence in *RPB* that originates from *PB* and has consequent in *Ext*(B). Because *CRPB* is completely quantified, by Lemma 41, *RPB* does not contain any p-sentence which originates from *PB* and has B as consequent. Because of the soundness of SLDNF, *RS* must not contain any p-sentence which originates from *PB*. Hence, S_0 is not deleted from *RS* by (S.2).

(2) In the second case S_0 originates from a p-sentence S in *PB*. There is a ground substitution θ_0 for S such that $prob(S\theta_0) = S_0$ and $context(S\theta_0)$ is a logical consequence of $CB \cup C$. There must be a θ^S, in fact θ^S is the restriction of θ_0 on variables in $cons(S)$, and, because of the completeness of SLDNF, θ^C as generated in BUILD-NET such that $context(S\theta_0)$ is a ground instance of $context(S\theta^S\theta^C)$. $ante(S\theta^S\theta^C)$ must be empty by induction hypothesis. Hence, $prob(S\theta^S\theta^C) = prob(S\theta^S)$ (because $prob(S\theta^S)$ is ground) $= S_0 \in RS$.

We prove $RS \subseteq \{S \mid S \in RPB \wedge cons(S) = B\}$: Assume $S_0 \in RS$. We need only to show that $S_0 \in RPB$. There exists $S \in$ *Extended_PB* and θ^S, θ^C such that $S_0 = prob(S\theta^S\theta^C)$, where θ^S and θ^C are the substitutions generated in BUILD-NET. There are two cases.

(1) In the first case S originates from *EBase*. *RS* must not contain any p-sentence which originates from *PB*. By Lemma 41 and because $RS \supseteq \{S \mid S \in RPB \wedge cons(S) = B\}$, there is no p-sentence in *RPB* which originates from *PB* and has consequent is in *Ext*(B). Hence, $S_0 \in RPB$ by the second step of the definition of *RPB* 9.

(2) In the second case S_0 originates from one sentence in *PB*. Because SLDNF is sound, $context(S\theta^S\theta^C)$ is a logical consequence of $CB \cup C$. Notice that $ante(S) = \{\}$. Hence, $cons(S_0) \in RAS$ and $S_0 \in RPB$.

(1.1.2) Follows from the fact that *RS* contains antecedent-less p-sentences.

Induction case $k = n$: Assume that the properties are correct for any execution of BUILD-NET which causes sequences of recursive calls on itself of depth $k < n$. We prove that they are also correct when the largest depth of those sequences is n.

(1.1.1) We prove $RS \supseteq \{S \mid S \in RPB \wedge cons(S) = B\}$: Assume $S_0 \in RPB$ and $cons(S_0) = B$. We will show $S_0 \in RS$. The proof when $ante(S_0) = \{\}$ is similar to that of the base case. We consider the case $ante(S_0) \neq \{\}$. S_0 must originate from a p-sentence S in *PB*. There is a ground substitution θ_0 for S such that $prob(S\theta_0) = S_0$ and $context(S\theta_0)$ is a logical consequence of $CB \cup C$. There must be a θ^S, in fact θ^S is the restriction of θ_0 on variables in $cons(S)$, and, because of the completeness of SLDNF, θ^C as generated in BUILD-NET such that $context(S\theta_0)$ is a ground instance of $context(S\theta^S\theta^C)$. There is a ground substitution θ_1 involving only variables in $context(S\theta^S\theta^C)$ such that $context(S\theta^S\theta^C\theta_1) = context(S\theta_0)$. The variables in S can be divided into three disjoint classes: the ones appearing in $cons(S)$, the ones appearing in $context(S)$ but not in $cons(S)$ and the rest. θ^S affects only variables in the first class and consistent with θ_0 on those variables (because θ^S is an mgu and B is ground).

$\theta^C\theta_1$ affects only variables in the second class and consistent with θ_0 on those variables (because θ^C is returned by SLDNF which uses only renamed variants of clauses in $CB \cup C$ by totally new variable names). We can form θ_2 which is the restriction of θ_0 on the variables in the last class. Hence, we have $prob(S\theta^S\theta^C\theta_1\theta_2) = prob(S\theta_0) = S_0$. $ante(S\theta^S\theta^C)$ is passed to BUILD-C-NET which makes calls to BUILD-NET. These invocations of BUILD-NET satisfy the induction hypothesis. By property (2.2), there is a substitution θ' returned by BUILD-C-NET such that $ante(S\theta^S\theta^C\theta') = ante(S\theta^S\theta^C\theta_1\theta_2)$. Hence $S_0 = prob(S\theta^S\theta^C\theta_1\theta_2) = prob(S\theta^S\theta^C\theta')$ is in RS.

We prove $RS \subseteq \{S \mid S \in RPB \wedge cons(S) = B\}$: Assume $S_0 \in RS$. We need only to show that $S_0 \in RPB$. The proof when $ante(S_0) = \{\}$ is similar to that of the base case. We consider the case $ante(S_0) \neq \{\}$. S_0 must originate from PB. There exists $S \in PB$ and $\theta^S, \theta^C, \theta'$ such that $S_0 = prob(S\theta^S\theta^C\theta')$, where θ^S, θ^C and θ' are the substitutions generated in BUILD-NET. Because SLDNF is sound, $context(S\theta^S\theta^C)$ is a logical consequence of $CB \cup C$, and so, any ground instance of it also has the same property. Notice that θ' is returned by BUILD-C-NET whose calls to BUILD-NET satisfy the induction hypothesis. Hence, by (2.1) any p-atom in $ante(S\theta^S\theta^C\theta')$ is in RAS. Hence, $cons(S_0) \in RAS$ and $S_0 \in RPB$.

(1.1.2) Follows from (1.3.3) and the induction hypothesis.

(P-BCN) The substitutions returned by BUILD-C-NET must be ground substitutions for the input p-atoms because the substitutions returned by BUILD-NET are ground substitutions for each input atom. We assume that the previous results (i.e. in (P-BN)) hold for any largest depth $\leqslant k$ and we prove that (2.1) and (2.2) hold if BUILD-C-NET does not cause any sequence of recursive calls on BUILD-NET of length greater than k. We prove by induction on the number of atoms in the conjunction. If $n = 1$ the result follow directly from property (1.3.3) of BUILD-NET. If $n > 1$, BUILD-C-NET works on the first $n-1$ input atoms before calling BUILD-NET with the input atom $A_n\theta$, where θ is one of the substitutions returned by BUILD-C-NET when the input conjunction is $A_1 \wedge \cdots \wedge A_{n-1}$. Because of (1.3.3), if BUILD-NET return θ' when the input atom is $A_n\theta$ then $A_n\theta\theta' \in RAS$. By induction hypothesis, $A_i\theta \in RAS$ and $A_i\theta$ is ground, $i = 1,\dots,n-1$. So, we have (2.1) because $A_i\theta\theta' = A_i\theta$, $i = 1,\dots,n-1$. Similarly, (2.2) holds by applying property (1.3.3). □

Proof of Proposition 35. We prove them in sequence:

(i) follows from (1.3.3) of Lemma 41.

(ii) follows directly from (1.3.2), (1.3.3), and the fact that $ground(E) \subseteq RAS$. □

Proof of Theorem 36. If SLDNF always stops then Q-procedure always stops for acyclic *PB*s with finite domains because in each pass of BUILD-NET the input atom is instantiated to ground and the function call to BUILD-C-NET transfers only atoms with smaller *P_ord*. The pruning step in the main procedure of Q-procedure does not affect the posterior probability of the queries [25]. The proof follows directly from the previous proposition, the property of supporting networks and the soundness of the Bayesian network updating procedure [23]. □

Proof of Theorem 38. Because the domains are finite and $CB \cup C$ is acyclic, any ground SLDNF-refutation has a bounded length [1]. A simple version of SLDNF in which every goal (or subgoal) is instantiated to ground (like in BUILD-NET) before the unification will stop on any input goal (by the same reason that makes BUILD-NET stops). That version of SLDNF is sound and complete for acyclic logic program with finite domains [1]. The result follows directly from Theorem 36. □

References

[1] K.R. Apt and M. Bezem, Acyclic programs, *New Generation Comput.* (1991) 335–363.

[2] F. Bacchus, Using first-order probability logic for the construction of Bayesian networks, in: *Proc. 9th Conf. on Uncertainty in Artificial Intelligence* (1993) 219–226.

[3] H.A. Blair and V.S. Subrahmanian, Paraconsistent logic programming, *Theoret. Comput. Sci.* (1987) 35–54.

[4] J.S. Breese, Construction of belief and decision networks, *Comput. Intelligence* **8** (1992) 624–647.

[5] G. F. Cooper, The computational complexity of probabilistic inference using bayesian belief networks, *Artificial Intelligence* **42** (1990) 393–405.

[6] F.J. Diez, Parameter adjustment in bayes networks. The generalized noisy or-gate, in: *Proc. 9th Conf. on Uncertainty in Artificial Intelligence*, Washington, DC (1993) 99–105.

[7] M.C. Fitting, Bilattices and the semantics of logic programming, *J. Logic Programming* (1988) 91–116.

[8] A.V. Gelder, K.A. Ross and J.S. Schlipf, The well-founded semantics for general logic programs, *J. ACM* (1991) 620–650.

[9] M. Gelfond and V. Liftschitz, The stable model semantics for logic programming, in: *Proc. of the 5th Internat. Conf. and Symposium on Logic Programming* (1988) 1070–1080.

[10] R.P. Goldman and E. Charniak, Dynamic construction of belief networks, in: *Proc. 6th Conf. on Uncertainty in Artificial Intelligence*, Cambridge, MA (1990) 90–97.

[11] R.P. Goldman and E. Charniak, A language for construction of belief networks, *IEEE Trans. Pattern Analysis Machine Intelligence* **15** (1993) 196–208.

[12] P. Haddawy, Generating Bayesian networks from probability logic knowledge bases, in: *Proc. 10th Conf. on Uncertainty in Artificial Intelligence*, Seattle (1994) 262–269.

[13] P. Haddawy, J.W. Helwig, L. Ngo and R. Krieger, Clinical simulation using context-sensitive temporal probability models, in: *Proc. 19th Ann. Symp. on Computer Applications in Medical Care (SCAMC95)*, New Orleans (1995) 203–207.

[14] D. Heckerman and M.P. Wellman, Bayesian networks, *Comm. ACM* **38** (1995) 27–30.

[15] M. Kifer and V.S. Subramahnian, Theory of generalized annotated logic programs and its applications, *J. Logic Programming* (1992) 335–367.

[16] D. Lin and R. Goebel, Integrating probabilistic, taxonomic and causal knowledge in abductive diagnosis, in: *Uncertainty in Artificial Intelligence: Vol. VI* (Elsevier, Amsterdam, 1991) 77–87.

[17] J.W. Lloyd, *Foundation of Logic Programming* (Springer, Berlin, 2nd ed., 1987).

[18] R. Ng, Semantics and consistency of empirical databases, in: *Proc. 1993 Internat. Conf. on Logic Programming* (1993) 812–826.

[19] R. Ng and V.S. Subrahmanian, A semantical framework for supporting subjective and conditional probability in deductive databases, in: *Proc. 1991 Internat. Conf. on Logic Programming* (1991) 565–580.

[20] R. Ng and V.S. Subrahmanian, Empirical probabilities in monadic deductive databases, in: *Proc. 8th Conf. on Uncertainty in Artificial Intelligence* (1992) 215–222.

[21] R. Ng and V.S. Subrahmanian, Probabilistic logic programming, *Inform. and Comput.* (1992) 150–201.

[22] L. Ngo, P. Haddawy and J. Helwig, A theoretical framework for context-sensitive temporal probability model construction with application to plan projection, in: *Proc. 11th Conf. on Uncertainty in Artificial Intelligence* (1995) 419–426.

[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, San Mateo, CA, 1988).

[24] D. Poole, Probabilistic horn abduction and bayesian networks, *Artificial Intelligence* **64** (1993) 81–129.
[25] R.D. Shachter, Probabilistic inference and influence diagrams, *Oper. Res.* **36** (1988) 589–604.
[26] S. Srinivas, A generalization of the noisy-or model, in: *Proc. 9th Conf. on Uncertainty in Artificial Intelligence* (1993) 208–217.
[27] S. Srinivas and J. Breese, IDEAL: A software package for analysis of influence diagrams, in: *Proc. 6th Conf. on Uncertainty in Artificial Intelligence* (1990) 212–219.
[28] M.H. van Emden, Quantitative deduction and its fixpoint theory, *J. Logic Programming* (1986) 37–53.
[29] M.P. Wellman, J.S. Breese and R.P. Goldman, From knowledge bases to decision models, *Knowledge Eng. Rev.* 7 (1992) 35–53.