Approximate Inference for Infinite Contingent Bayesian Networks

Brian Milch, Bhaskara Marthi, David Sontag, Stuart Russell, Daniel L. Ong and Andrey Kolobov
Computer Science Division
University of California
Berkeley, CA 94720-1776
{milch,bhaskara,russell,dsontag,dlong,karaya1}@cs.berkeley.edu

Abstract

In many practical problems—from tracking aircraft based on radar data to building a bibliographic database based on citation lists—we want to reason about an unbounded number of unseen objects with unknown relations among them. Bayesian networks, which define a fixed dependency structure on a finite set of variables, are not the ideal representation language for this task. This paper introduces *contingent Bayesian networks* (CBNs), which represent uncertainty about dependencies by labeling each edge with a condition under which it is active. A CBN may contain cycles and have infinitely many variables. Nevertheless, we give general conditions under which such a CBN defines a unique joint distribution over its variables. We also present a likelihood weighting algorithm that performs approximate inference in finite time per sampling step on any CBN that satisfies these conditions.

1 Introduction

One of the central tasks an intelligent agent must perform is to make inferences about the real-world objects that underlie its observations. This type of reasoning has a wide range of practical applications, from tracking aircraft based on radar data to building a bibliographic database based on citation lists. To tackle these problems, it makes sense to use probabilistic models that represent uncertainty about the number of underlying objects, the relations among them, and the mapping from observations to objects.

Over the past decade, a number of probabilistic modeling formalisms have been developed that explicitly represent objects and relations. Most work has focused on scenarios where, for any given query, there is no uncertainty about the set of relevant objects. In extending this line of work to unknown sets of objects, we face a difficulty: unless we place an upper bound on the number of underlying objects, the resulting model has infinitely many variables. We have developed a formalism called BLOG (Bayesian LOGic) in which such infinite models can be defined concisely [7]. However, it is not obvious under what conditions such models define probability distributions, or how to do inference on them.

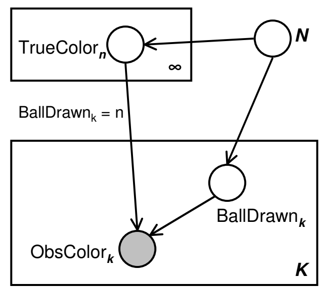

Figure 1: A graphical model (with plates representing repeated elements) for the balls-and-urn example. This is a BN if we disregard the labels $\mathsf{BallDrawn}_k = n$ on the edges $\mathsf{TrueColor}_n \rightarrow \mathsf{ObsColor}_k$ for $k \in \{1, \ldots, K\}$, $n \in \{1, 2, \ldots\}$. With the labels, it is a CBN.

Bayesian networks (BNs) with infinitely many variables are actually quite common: for instance, a dynamic BN with time running infinitely into the future has infinitely many nodes. These common models have the property that each node has only finitely many ancestors. So for finite sets of evidence and query variables, pruning away "barren" nodes [15] yields a finite BN that is sufficient for answering the query. However, generative probability models with unknown objects often involve infinite ancestor sets, as illustrated by the following stylized example from [13].

Example 1. *Suppose an urn contains some unknown number of balls N, and suppose our prior distribution for N assigns positive probability to every natural number. Each ball has a color—say, black or white—chosen independently from a fixed prior. Suppose we repeatedly draw a ball uniformly at random, observe its color, and return it to the urn. We cannot distinguish two identically colored balls from each other. Furthermore, we have some (known) probability of making a mistake in each color observation.*

Given our observations, we might want to predict the total number of balls in the urn, or solve the identity uncertainty *problem: computing the posterior probability that (for example) we drew the same ball on our first two draws.*

Fig. 1 shows a graphical model for this example. There is an infinite set of variables for the true colors of the balls; each $\mathsf{TrueColor}_n$ variable takes the special value null when $N < n$. Each $\mathsf{BallDrawn}_k$ variable takes a value between 1 and N, indicating the ball drawn on draw k. The $\mathsf{ObsColor}_k$ variable then depends on $\mathsf{TrueColor}_{(\mathsf{BallDrawn}_k)}$. In this BN, all the infinitely many $\mathsf{TrueColor}_n$ variables are ancestors of each $\mathsf{ObsColor}_k$ variable. Thus, even if we prune barren nodes, we cannot obtain a finite BN for computing the posterior over N. The same problem arises in real-world identity uncertainty tasks, such as resolving coreference among citations that refer to some underlying publications [10].

Bayesian networks also fall short in representing scenarios where the relations between objects or events—and thus the dependencies between random variables—are random.

Example 2. *Suppose a hurricane is going to strike two cities, Alphatown and Betaville, but it is not known which city will be hit first. The amount of damage in each city depends on the level of preparations made in each city. Also, the level of preparations in the second city depends on the amount of damage in the first city. Fig. 2 shows a model for this situation, where the variable F takes on the value A or B to indicate whether Alphatown or Betaville is hit first.*

In this example, suppose that we have a good estimate of the distribution for preparations in the first city, and of the conditional probability distribution (CPD) for preparations in the second city given damage in the first. The obvious graphical model to draw is the one in Fig. 2, but it has a figure-eight-shaped cycle. Of course, we can construct a BN for the intended distribution by choosing an arbitrary ordering of the variables and including all necessary edges to each variable from its predecessors. Suppose we use the ordering F, P_A, D_A, P_B, D_B. Then $P(P_A|F=A)$ is easy to write down, but to compute $P(P_A|F=B)$ we need to sum out P_B and D_B. There is no acyclic BN that reflects our causal intuitions.

Using a high-level modeling language, one can represent scenarios such as those in Figs. 1 and 2 in a compact and natural way. However, as we have seen, the BNs corresponding to such models may contain cycles or infinite ancestor sets. The assumptions of finiteness and acyclicity are fundamental not just for BN inference algorithms, but also for the standard theorem that every BN defines a unique joint distribution.

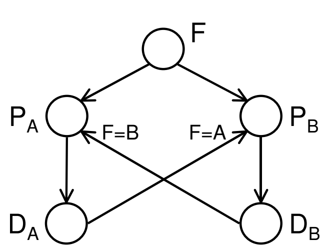

Figure 2: A cyclic BN for the hurricane scenario. P stands for preparations, D for damage, A for Alphatown, B for Betaville, and F for the city that is hit first.

Our approach to such models is based on the notion of context-specific independence (CSI) [1]. In the balls-and-urn example, in the context $\mathsf{BallDrawn}_k = n$, $\mathsf{ObsColor}_k$ has only one other ancestor — $\mathsf{TrueColor}_n$. Similarly, the BN in Fig. 2 is acyclic in the context $F = A$ and also in the context $F = B$. To exploit these CSI properties, we define two generalizations of BNs that make CSI explicit. The first is *partition-based models* (PBMs), where instead of specifying a set of parents for each variable, one specifies an arbitrary partition of the outcome space that determines the variable's CPD. In Sec. 2, we give an abstract criterion that guarantees that a PBM defines a unique joint distribution.

To prove more concrete results, we focus in Sec. 3 on the special case of *contingent Bayesian networks* (CBNs): possibly infinite BNs where some edges are labeled with conditions. CBNs combine the use of decision trees for CPDs [1] with the idea of labeling edges to indicate when they are active [3]. In Sec. 3, we provide general conditions under which a contingent BN defines a unique probability distribution, even in the presence of cycles or infinite ancestor sets. In Sec. 4 we explore the extent to which results about CBNs carry over to the more general PBMs. Then in Sec. 5 we present a sampling algorithm for approximate inference in contingent BNs. The time required to generate a sample using this algorithm depends only on the size of the context-specifically relevant network, not the total size of the CBN (which may be infinite). Experimental results for this algorithm are given in Sec. 6. We omit proofs for reasons of space; the proofs can be found in our technical report [8].

2 Partition-based models

We assume a set $\mathcal{V}$ of random variables, which may be countably infinite. Each variable X has a domain $\mathrm{dom}(X)$; we assume in this paper that each domain is at most countably infinite. The outcome space over which we would like to define a probability measure is the product space $\Omega \triangleq \times_{(X \in \mathcal{V})} \mathrm{dom}(X)$. An outcome $\omega \in \Omega$ is an assignment of values to all the variables; we write $X(\omega)$ for the value of X in ω.

An *instantiation* σ is an assignment of values to a subset of $\mathcal{V}$. We write $\mathrm{vars}(\sigma)$ for the set of variables to which σ assigns values, and σ_X for the value that σ assigns to a variable $X \in \mathrm{vars}(\sigma)$. The empty instantiation is denoted $\varnothing$. An instantiation σ is said to be finite if $\mathrm{vars}(\sigma)$ is finite. The *completions* of σ, denoted $\mathrm{comp}(\sigma)$, are those

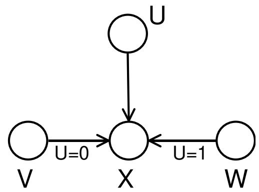

Figure 3: A simple contingent BN.

outcomes that agree with σ on $\mathrm{vars}(\sigma)$:

$$\mathrm{comp}(\sigma) \triangleq \{\omega \in \Omega \, : \, \forall X \in \mathrm{vars}(\sigma), X(\omega) = \sigma_X\}$$

If σ is a *full* instantiation — that is, $\mathrm{vars}(\sigma) = \mathcal{V}$ — then $\mathrm{comp}(\sigma)$ consists of just a single outcome.

To motivate our approach to defining a probability measure on Ω, consider the BN in Fig. 3, ignoring for now the labels on the edges. To completely specify this model, we would have to provide, in addition to the graph structure, a conditional probability distribution (CPD) for each variable. For example, assuming the variables are binary, the CPD for X would be a table with 8 rows, each corresponding to an instantiation of X's three parents. Another way of viewing this is that X's parent set defines a partition of Ω where each CPT row corresponds to a block (i.e., element) of the partition. This may seem like a pedantic rephrasing, but partitions can expose more structure in the CPD. For example, suppose X depends only on V when $U=0$ and only on W when $U=1$. The tabular CPD for X would still be the same size, but now the partition for X only has four blocks: $\mathrm{comp}(U=0, V=0)$, $\mathrm{comp}(U=0, V=1)$, $\mathrm{comp}(U=1, W=0)$, and $\mathrm{comp}(U=1, W=1)$.

Definition 1. *A* partition-based model Γ *over $\mathcal{V}$ consists of*

- *for each $X \in \mathcal{V}$, a partition Λ_X^Γ of Ω where we write $\lambda_X^\Gamma(\omega)$ to denote the block of the partition that the outcome ω belongs to;*
- *for each $X \in \mathcal{V}$ and block $\lambda \in \Lambda_X^\Gamma$, a probability distribution $p_\Gamma(X|\lambda)$ over $\mathrm{dom}(X)$.*

A PBM defines a probability distribution over Ω. If $\mathcal{V}$ is finite, this distribution can be specified as a product expression, just as for an ordinary BN:

$$P(\omega) \triangleq \prod_{X \in \mathcal{V}} p_\Gamma(X(\omega)|\lambda_X^\Gamma(\omega)) \qquad (1)$$

Unfortunately, this equation becomes meaningless when $\mathcal{V}$ is infinite, because the probability of each outcome ω will typically be zero. A natural solution is to define the probabilities of finite instantiations, and then rely on Kolmogorov's extension theorem (see, e.g., [2]) to ensure that we have defined a unique distribution over outcomes. But Eq. 1 relies on having a full outcome ω to determine which CPD to use for each variable X.

How can we write a similar product expression that involves only a partial instantiation? We need the notion of a partial instantiation *supporting* a variable.

Definition 2. *In a PBM Γ, an instantiation σ* supports *a variable X if there is some block $\lambda \in \Lambda_X^\Gamma$ such that $\mathrm{comp}(\sigma) \subseteq \lambda$. In this case we write $\lambda_X^\Gamma(\sigma)$ for the unique element of Λ_X^Γ that has $\mathrm{comp}(\sigma)$ as a subset.*

Intuitively, σ supports X if knowing σ is enough to tell us which block of Λ_X^Γ we're in, and thus which CPD to use for X. In Fig. 3, $(U=0, V=0)$ supports X, but $(U=1, V=0)$ does not. In an ordinary BN, any instantiation of the parents of X supports X.

An instantiation σ is *self-supporting* if every $X \in \mathrm{vars}(\sigma)$ is supported by σ. In a BN, if $\mathbf{U}$ is an ancestral set (a set of variables that includes all the ancestors of its elements), then every instantiation of $\mathbf{U}$ is self-supporting.

Definition 3. *A probability measure P over $\mathcal{V}$* satisfies *a PBM Γ if for every finite, self-supporting instantiation σ:*

$$P(\mathrm{comp}(\sigma)) = \prod_{X \in \mathrm{vars}(\sigma)} p_\Gamma(\sigma_X|\lambda_X^\Gamma(\sigma)) \qquad (2)$$

A PBM is *well-defined* if there is a unique probability measure that satisfies it. One way a PBM can fail to be well-defined is if the constraints specified by Eq. 2 are inconsistent: for instance, if they require that the instantiations $(X=1, Y=1)$ and $(X=0, Y=0)$ both have probability 0.9. Conversely, a PBM can be satisfied by many distributions if, for example, the only self-supporting instantiations are infinite ones — then Def. 3 imposes no constraints.

When can we be sure that a PBM is well-defined? First, recall that a BN is well-defined if it is acyclic, or equivalently, if its nodes have a topological ordering. Thus, it seems reasonable to think about numbering the variables in a PBM. A *numbering* of $\mathcal{V}$ is a bijection π from $\mathcal{V}$ to some prefix of $\mathbb{N}$ (this will be a proper prefix if $\mathcal{V}$ is finite, and the whole set $\mathbb{N}$ if $\mathcal{V}$ is countably infinite). We define the predecessors of a variable X under π as:

$$\mathrm{Pred}_\pi[X] \triangleq \{U \in \mathcal{V} \, : \, \pi(U) < \pi(X)\}$$

Note that since each variable X is assigned a finite number $\pi(X)$, the predecessor set $\mathrm{Pred}_\pi[X]$ is always finite.

One of the purposes of PBMs is to handle cyclic scenarios such as Example 2. Thus, rather than speaking of a single topological numbering for a model, we speak of a *supportive numbering* for each outcome.

Definition 4. *A numbering π is a* supportive numbering *for an outcome ω if for each $X \in \mathcal{V}$, the instantiation $\mathrm{Pred}_\pi[X](\omega)$ supports X.*

Theorem 1. *A PBM Γ is well-defined if, for every outcome $\omega \in \Omega$, there exists a supportive numbering π_ω.*

The converse of this theorem is not true: a PBM may happen to be well-defined even if some outcomes do not have supportive numberings. But more importantly, the requirement that each outcome have a supportive numbering is very abstract. How could we determine whether it holds for a given PBM? To answer this question, we now turn to a more concrete type of model.

3 Contingent Bayesian networks

Contingent Bayesian networks (CBNs) are a special case of PBMs for which we can define more concrete well-definedness criteria, as well as an inference algorithm. In Fig. 3 the partition was represented not as a list of blocks, but implicitly by labeling each edge with an event. The meaning of an edge from W to X labeled with an event E, which we denote by $(W \to X \mid E)$, is that the value of W may be relevant to the CPD for X only when E occurs. In Fig. 3, W is relevant to X only when $U = 1$.

Using the definitions of $\mathcal{V}$ and Ω from the previous section, we can define a CBN structure as follows:

Definition 5. *A* CBN structure *$\mathcal{G}$ is a directed graph where the nodes are elements of $\mathcal{V}$ and each edge is labeled with a subset of Ω.*

In our diagrams, we leave an edge blank when it is labeled with the uninformative event Ω. An edge $(W \to X \mid E)$ is said to be *active* given an outcome ω if $\omega \in E$, and active given a partial instantiation σ if $\mathrm{comp}(\sigma) \subseteq E$. A variable W is an *active parent* of X given σ if an edge from W to X is active given σ.

Just as a BN is parameterized by specifying CPTs, a CBN is parameterized by specifying a decision tree for each node.

Definition 6. *A* decision tree *$\mathcal{T}$ is a directed tree where each node is an instantiation σ, such that:*

- *the root node is $\varnothing$;*
- *each non-leaf node σ* splits on *a variable $X^{\mathcal{T}}_{\sigma}$ such that the children of σ are $\{(\sigma; X^{\mathcal{T}}_{\sigma} = x) \ : \ x \in \mathrm{dom}\left(X^{\mathcal{T}}_{\sigma}\right)\}$.*

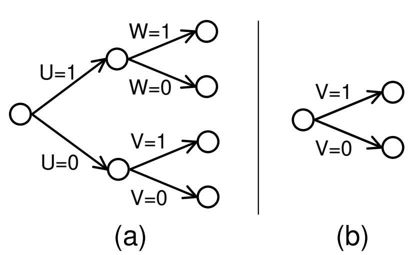

Figure 4: Two decision trees for X in Fig. 3. Tree (a) respects the CBN structure, while tree (b) does not.

Two decision trees are shown in Fig. 4. If a node splits on a variable that has infinitely many values, then it will have infinitely many children. This definition also allows a decision tree to contain infinite paths. However, each node in the tree is a finite instantiation, since it is connected to the root by a finite path. We will call a path *truncated* if it ends with a non-leaf node. Thus, a non-truncated path either continues infinitely or ends at a leaf. An outcome ω *matches* a path θ if ω is a completion of every node (instantiation) in the path. The non-truncated paths starting from the root are mutually exclusive and exhaustive, so a decision tree defines a partition of Ω.

Definition 7. *The partition $\Lambda_{\mathcal{T}}$ defined by a decision tree $\mathcal{T}$ consists of a block of the form $\{\omega \in \Omega \ : \ \omega \text{ matches } \theta\}$ for each non-truncated path θ starting at the root of $\mathcal{T}$.*

So for each variable X, we specify a decision tree $\mathcal{T}_X$, thus defining a partition $\Lambda_X \triangleq \Lambda_{(\mathcal{T}_X)}$. To complete the parameterization, we also specify a function $p_{\mathcal{B}}(X = x|\lambda)$ that maps each $\lambda \in \Lambda_X$ to a distribution over $\mathrm{dom}\,(X)$. However, the decision tree for X must respect the CBN structure in the following sense.

Definition 8. *A decision tree $\mathcal{T}$* respects the CBN structure *$\mathcal{G}$ at X if for every node $\sigma \in \mathcal{T}$ that splits on a variable W, there is an edge $(W \to X \mid E)$ in $\mathcal{G}$ that is active given σ.*

For example, tree (a) in Fig. 4 respects the CBN structure of Fig. 3 at X. However, tree (b) does not: the root instantiation $\varnothing$ does not activate the edge $(V \to X \mid U = 0)$, so it should not split on V.

Definition 9. *A* contingent Bayesian network *(CBN) $\mathcal{B}$ over $\mathcal{V}$ consists of a CBN structure $\mathcal{G}^{\mathcal{B}}$, and for each variable $X \in \mathcal{V}$:*

- *a decision tree $\mathcal{T}^{\mathcal{B}}_X$ that respects $\mathcal{G}^{\mathcal{B}}$ at X, defining a partition $\Lambda^{\mathcal{B}}_X \triangleq \Lambda_{(\mathcal{T}^{\mathcal{B}}_X)}$;*
- *for each block $\lambda \in \Lambda^{\mathcal{B}}_X$, a probability distribution $p_{\mathcal{B}}(X|\lambda)$ over $\mathrm{dom}\,(X)$.*

It is clear that a CBN is a kind of PBM, since it defines a partition and a conditional probability distribution for each variable. Thus, we can carry over the definitions from the previous section of what it means for a distribution to satisfy a CBN, and for a CBN to be well-defined.

We will now give a set of structural conditions that ensure that a CBN is well-defined. We call a set of edges in $\mathcal{G}$ *consistent* if the events on the edges have a non-empty intersection: that is, if there is some outcome that makes all the edges active.

Theorem 2. *Suppose a CBN $\mathcal{B}$ satisfies the following:*

(A1) No consistent path in $\mathcal{G}^{\mathcal{B}}$ forms a cycle.
(A2) No consistent path in $\mathcal{G}^{\mathcal{B}}$ forms an infinite receding chain $X_1 \leftarrow X_2 \leftarrow X_3 \leftarrow \cdots$.
(A3) No variable $X \in \mathcal{V}$ has an infinite, consistent set of incoming edges in $\mathcal{G}^{\mathcal{B}}$.

Then $\mathcal{B}$ is well-defined.

A CBN that satisfies the conditions of Thm. 2 is said to be *structurally well-defined*. If a CBN has a finite set of variables, we can check the conditions directly. For instance, the CBN in Fig. 2 is structurally well-defined: although it contains a cycle, the cycle is not consistent.

The balls-and-urn example (Fig. 1) has infinitely many nodes, so we cannot write out the CBN explicitly. However, it is clear from the plates representation that this CBN is structurally well-defined as well: there are no cycles or infinite receding chains, and although each $\mathsf{ObsColor}_k$ node has infinitely many incoming edges, the labels $\mathsf{BallDrawn}_k = n$ ensure that exactly one of these edges is active in each outcome. In [8], we discuss the general problem of determining whether the infinite CBN defined by a high-level model is structurally well-defined.

4 CBNs as implementations of PBMs

In a PBM, we specify an arbitrary partition for each variable; in CBNs, we restrict ourselves to partitions generated by decision trees. But given any partition Λ, we can construct a decision tree $\mathcal{T}$ that yields a partition at least as fine as Λ—that is, such that each block $\lambda \in \Lambda_{\mathcal{T}}$ is a subset of some $\lambda' \in \Lambda$. In the worst case, every path starting at the root in $\mathcal{T}$ will need to split on every variable. Thus, every PBM is *implemented* by some CBN, in the following sense:

Definition 10. *A CBN $\mathcal{B}$* implements *a PBM Γ over the same set of variables $\mathcal{V}$ if, for each variable $X \in \mathcal{V}$, each block $\lambda \in \Lambda_X^{\mathcal{B}}$ is a subset of some block $\lambda' \in \Lambda_X^{\Gamma}$, and $p_{\mathcal{B}}(X|\lambda) = p_{\Gamma}(X|\lambda')$.*

Theorem 3. *If a CBN $\mathcal{B}$ implements a PBM Γ and $\mathcal{B}$ is structurally well-defined, then Γ is also well-defined, and $\mathcal{B}$ and Γ are satisfied by the same unique distribution.*

Thm. 3 gives us a way to show that a PBM Γ is well-defined: construct a CBN $\mathcal{B}$ that implements Γ, and then use Thm. 2 to show that $\mathcal{B}$ is structurally well-defined. However, the following example illustrates a complication:

Example 3. *Consider predicting who will go to a weekly book group meeting. Suppose it is usually Bob's responsibility to prepare questions for discussion, but if a historical fiction book is being discussed, then Alice prepares questions. In general, Alice and Bob each go to the meeting with probability 0.9. However, if the book is historical fiction and Alice isn't going, then the group will have no discussion questions, so the probability that Bob bothers to go is only 0.1. Similary, if the book is not historical fiction and Bob isn't going, then Alice's probability of going is 0.1. We will use H, G_A and G_B to represent the binary variables "historical fiction", "Alice goes", and "Bob goes".*

This scenario is most naturally represented by a PBM. The probability that Bob goes is 0.1 given $((H=1) \wedge (G_A=0))$ and 0.9 otherwise, so the partition for G_B has two blocks. The partition for G_A has two blocks as well.

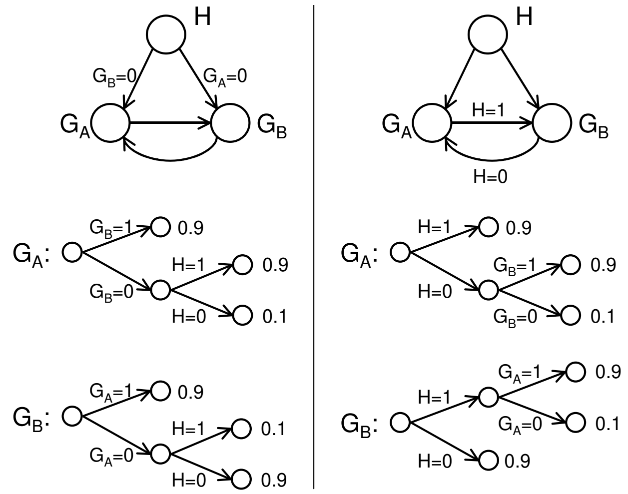

Figure 5: Two CBNs for Ex. 3, with decision trees and probabilities for G_A and G_B.

The CBNs in Fig. 5 both implement this PBM. There are no decision trees that yield exactly the desired partitions for G_A and G_B: the trees in Fig. 5 yield three blocks instead of two. Because the trees on the two sides of the figure split on the variables in different orders, they respect CBN structures with different labels on the edges. The CBN on the left has a consistent cycle, while the CBN on the right is structurally well-defined.

Thus, there may be multiple CBNs that implement a given PBM, and it may be that some of these CBNs are structurally well-defined while others are not. Even if we are given a well-defined PBM, it may be non-trivial to find a structurally well-defined CBN that implements it. Thus, algorithms that apply to structurally well-defined CBNs — such as the one we define in the next section — cannot be extended easily to general PBMs.

5 Inference

In this section we discuss an approximate inference algorithm for CBNs. To get information about a given CBN $\mathcal{B}$, our algorithm will use a few "black box" oracle functions. The function GET-ACTIVE-PARENT(X, σ) returns a variable that is an active parent of X given σ but is not already included in $\mathrm{vars}(\sigma)$. It does this by traversing the decision tree $\mathcal{T}_X^{\mathcal{B}}$, taking the branch associated with σ_U when the tree splits on a variable $U \in \mathrm{vars}(\sigma)$, until it reaches a split on a variable not included in $\mathrm{vars}(\sigma)$. If there is no such variable — which means that σ supports X — then it returns null. We also need the function COND-PROB(X, x, σ), which returns $p_{\mathcal{B}}(X=x|\sigma)$ whenever σ supports X, and the function SAMPLE-VALUE(X, σ), which randomly samples a value according to $p_{\mathcal{B}}(X|\sigma)$.

Our inference algorithm is a form of likelihood weight-

function CBN-LIKELIHOOD-WEIGHTING($\mathbf{Q}, \mathbf{e}, \mathcal{B}, N$)
  returns an estimate of $P(\mathbf{Q}|\mathbf{e})$
  inputs: $\mathbf{Q}$, the set of query variables
    $\mathbf{e}$, evidence specified as an instantiation
    $\mathcal{B}$, a contingent Bayesian network
    N, the number of samples to be generated

  $\mathbf{W} \leftarrow$ a map from $\mathrm{dom}(\mathbf{Q})$ to real numbers, with values lazily initialized to zero when accessed
  for j = 1 to N **do**
    $\sigma, w \leftarrow$ CBN-WEIGHTED-SAMPLE($\mathbf{Q}, \mathbf{e}, \mathcal{B}$)
    $\mathbf{W}[\mathbf{q}] \leftarrow \mathbf{W}[\mathbf{q}] + w$ where $\mathbf{q} = \sigma_{\mathbf{Q}}$
  return NORMALIZE($\mathbf{W}[\mathbf{Q}]$)

---

function CBN-WEIGHTED-SAMPLE($\mathbf{Q}, \mathbf{e}, \mathcal{B}$)
  returns an instantiation and a weight

  $\sigma \leftarrow \varnothing$; $stack \leftarrow$ an empty stack; $w \leftarrow 1$
  loop
    if $stack$ is empty
      if some X in ($\mathbf{Q} \cup \mathrm{vars}(\mathbf{e})$) is not in $\mathrm{vars}(\sigma)$
        PUSH(X, $stack$)
      else
        return σ, w

    while X on top of $stack$ is not supported by σ
      $V \leftarrow$ GET-ACTIVE-PARENT(X, σ)
      push V on $stack$

    $X \leftarrow$ POP($stack$)
    if X in $\mathrm{vars}(\mathbf{e})$
      $x \leftarrow \mathbf{e}_X$
      $w \leftarrow w \times$ COND-PROB(X, x, σ)
    else
      $x \leftarrow$ SAMPLE-VALUE(X, σ)
    $\sigma \leftarrow (\sigma,\ X = x)$

Figure 6: Likelihood weighting algorithm for CBNs.

ing. Recall that the likelihood weighting algorithm for BNs samples all non-evidence variables in topological order, then weights each sample by the conditional probability of the observed evidence [14]. Of course, we cannot sample all the variables in an infinite CBN. But even in a BN, it is not necessary to sample *all* the variables: the relevant variables can be found by following edges backwards from the query and evidence variables. We extend this notion to CBNs by only following edges that are active given the instantiation sampled so far. At each point in the algorithm (Fig. 6), we maintain an instantiation σ and a stack of variables that need to be sampled. If the variable X on the top of the stack is supported by σ, we pop X off the stack and sample it. Otherwise, we find a variable V that is an active parent of X given σ, and push V onto the stack. If the CBN is structurally admissible, this process terminates in finite time: condition (A1) ensures that we never push the same variable onto the stack twice, and conditions (A2) and (A3) ensure that the number of distinct variables pushed onto the stack is finite.

As an example, consider the balls-and-urn CBN (Fig. 1). If we want to query N given some color observations, the algorithm begins by pushing N onto the stack. Since N (which has no parents) is supported by $\varnothing$, it is immediately removed from the stack and sampled. Next, the first evidence variable $\mathsf{ObsColor}_1$ is pushed onto the stack. The active edge into $\mathsf{ObsColor}_1$ from $\mathsf{BallDrawn}_1$ is traversed, and $\mathsf{BallDrawn}_1$ is sampled immediately because it is supported by σ (which now includes N). The edge from $\mathsf{TrueColor}_n$ (for n equal to the sampled value of $\mathsf{BallDrawn}_1$) to $\mathsf{ObsColor}_1$ is now active, and so $\mathsf{TrueColor}_n$ is sampled as well. Now $\mathsf{ObsColor}_1$ is finally supported by σ, so it is removed from the stack and instantiated to its observed value. This process is repeated for all the observations. The resulting sample will get a high weight if the sampled true colors for the balls match the observed colors.

Intuitively, this algorithm is the same as likelihood weighting, in that we sample the variables in some topological order. The difference is that we sample only those variables that are needed to support the query and evidence variables, and we do not bother sampling any of the other variables in the CBN. Since the weight for a sample only depends on the conditional probabilities of the evidence variables, sampling additional variables would have no effect.

Theorem 4. *Given a structurally well-defined CBN $\mathcal{B}$, a finite evidence instantiation* $\mathbf{e}$*, a finite set* $\mathbf{Q}$ *of query variables, and a number of samples N, the algorithm* CBN-LIKELIHOOD-WEIGHTING *in Fig. 6 returns an estimate of the posterior distribution $P(\mathbf{Q}|\mathbf{e})$ that converges with probability 1 to the correct posterior as $N \to \infty$. Furthermore, each sampling step takes a finite amount of time.*

6 Experiments

We ran two sets of experiments using the likelihood weighting algorithm of Fig. 6. Both use the balls and urn setup from Ex. 1. The first experiment estimates the number of balls in the urn given the colors observed on 10 draws; the second experiment is an identity uncertainty problem. In both cases, we run experiments with both a noiseless sensor model, where the observed colors of balls always match their true colors, and a noisy sensor model, where with probability 0.2 the wrong color is reported.

The purpose of these experiments is to show that inference over an infinite number of variables can be done using a general algorithm in finite time. We show convergence of our results to the correct values, which were computed by enumerating equivalence classes of outcomes with up to 100 balls (see [8] for details). More efficient sampling algorithms for these problems have been designed by hand [9]; however, our algorithm is general-purpose, so it needs no modification to be applied to a different domain.

Number of balls: In the first experiment, we are predicting the total number of balls in the urn. The prior over the number of balls is a Poisson distribution with mean 6; each

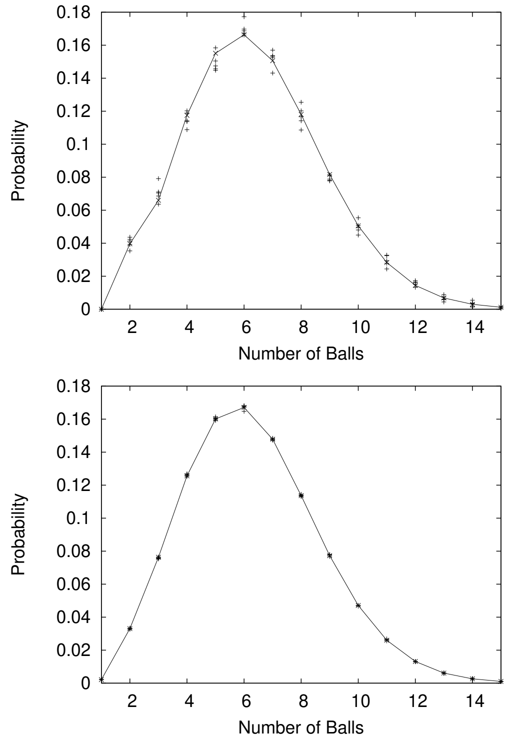

Figure 7: Posterior distributions for the total number of balls given 10 observations in the noise-free case (top) and noisy case (bottom). Exact probabilities are denoted by '×'s and connected with a line; estimates from 5 sampling runs are marked with '+'s.

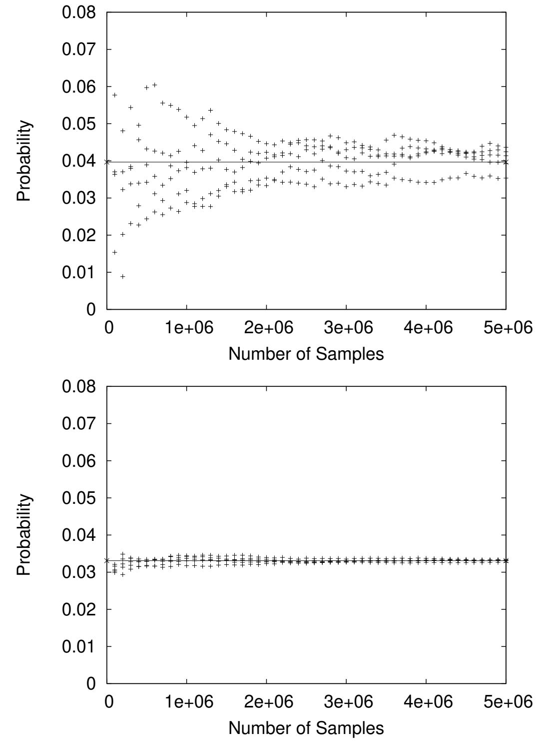

Figure 8: Probability that $N = 2$ given 10 observations (5 black, 5 white) in the noise-free case (top) and noisy case (bottom). Solid line indicates exact value; '+'s are values computed by 5 sampling runs at intervals of 100,000 samples.

ball is black with probability 0.5. The evidence consists of color observations for 10 draws from the urn: five are black and five are white. For each observation model, five independent trials were run, each of 5 million samples.[1]

Fig. 7 shows the posterior probabilities for total numbers of balls from 1 to 15 computed in each of the five trials, along with the exact probabilities. The results are all quite close to the true probability, especially in the noisy-observation case. The variance is higher for the noise-free model because the sampled true colors for the balls are often inconsistent with the observed colors, so many samples have zero weights.

Fig. 8 shows how quickly our algorithm converges to the correct value for a particular probability, $P(N = 2|\text{obs})$. The run with deterministic observations stays within 0.01 of the true probability after 2 million samples. The noisy-observation run converges faster, in just 100,000 samples.

Identity uncertainty: In the second experiment, three balls are drawn from the urn: a black one and then two white ones. We wish to find the probability that the second and third draws produced the same ball. The prior distribution over the number of balls is Poisson(6). Unlike the previous experiment, each ball is black with probability 0.3.

We ran five independent trials of 100,000 samples on the deterministic and noisy observation models. Fig. 9 shows the estimates from all five trials approaching the true probability as the number of samples increases. Note that again, the approximations for the noisy observation model converge more quickly. The noise-free case stays within 0.01 of the true probability after 70,000 samples, while the noisy case converges within 10,000 samples. Thus, we perform inference over a model with an unbounded number of objects and get reasonable approximations in finite time.

7 Related work

There are a number of formalisms for representing context-specific independence (CSI) in BNs. Boutilier et al. [1] use decision trees, just as we do in CBNs. Poole and Zhang [12] use a set of *parent contexts* (partial instantiations of the parents) for each node; such models can be represented as PBMs, although not necessarily as CBNs. Neither paper discusses infinite or cyclic models. The idea of labeling edges with the conditions under which they are active may have originated in [3] (a working paper that is no longer available); it was recently revived in [5].

[1]Our Java implementation averages about 1700 samples/sec. for the exact observation case and 1100 samples/sec. for the noisy observation model on a 3.2 GHz Intel Pentium 4.

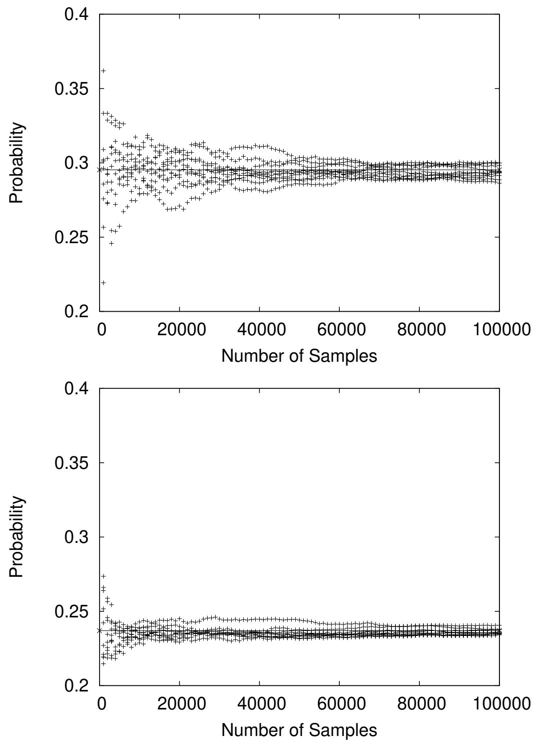

Figure 9: Probability that draws two and three produced the same ball for noise-free observations (top) and noisy observations (bottom). Solid line indicates exact value; '+'s are values computed by 5 sampling runs.

Bayesian multinets [4] can represent models that would be cyclic if they were drawn as ordinary BNs. A multinet is a mixture of BNs: to sample an outcome from a multinet, one first samples a value for the *hypothesis variable* H, and then samples the remaining variables using a hypothesis-specific BN. We could extend this approach to CBNs, representing a structurally well-defined CBN as a (possibly infinite) mixture of acyclic, finite-ancestor-set BNs. However, the number of hypothesis-specific BNs required would often be exponential in the number of variables that govern the dependency structure. On the other hand, to represent a given multinet as a CBN, we simply include an edge $V \rightarrow X$ with the label $H = h$ whenever that edge is present in the hypothesis-specific BN for h.

There has also been some work on handling infinite ancestor sets in BNs without representing CSI. Jaeger [6] states that an infinite BN defines a unique distribution if there is a well-founded topological ordering on its variables; that condition is more complete than ours in that it allows a node to have infinitely many active parents, but less complete in that it requires a single ordering for all contexts. Pfeffer and Koller [11] point out that a network containing an infinite receding path $X_1 \leftarrow X_2 \leftarrow X_3 \leftarrow \cdots$ may still define a unique distribution if the CPDs along the path form a Markov chain with a unique stationary distribution.

8 Conclusion

We have presented contingent Bayesian networks, a formalism for defining probability distributions over possibly infinite sets of random variables in a way that makes context-specific independence explicit. We gave structural conditions under which a CBN is guaranteed to define a unique distribution—even if it contains cycles, or if some variables have infinite ancestor sets. We presented a sampling algorithm that is guaranteed to complete each sampling step in finite time and converge to the correct posterior distribution. We have also discussed how CBNs fit into the more general framework of partition-based models.

Our likelihood weighting algorithm, while completely general, is not efficient enough for most real-world problems. Our future work includes developing an efficient Metropolis-Hastings sampler that allows for user-specified proposal distributions; the results of [10] suggest that such a system can handle large inference problems satisfactorily. Further work at the theoretical level includes handling continuous variables, and deriving more complete conditions under which CBNs are guaranteed to be well-defined.

References

[1] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proc. 12th UAI*, pages 115–123, 1996.

[2] R. Durrett. *Probability: Theory and Examples*. Wadsworth, Belmont, CA, 2nd edition, 1996.

[3] R. M. Fung and R. D. Shachter. Contingent influence diagrams. Working Paper, Dept. of Engineering-Economic Systems, Stanford University, 1990.

[4] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *AIJ*, 82(1–2):45–74, 1996.

[5] D. Heckerman, C. Meek, and D. Koller. Probabilistic models for relational data. Technical Report MSR-TR-2004-30, Microsoft Research, 2004.

[6] M. Jaeger. Reasoning about infinite random structures with relational Bayesian networks. In *Proc. 6th KR*, 1998.

[7] B. Milch, B. Marthi, and S. Russell. BLOG: Relational modeling with unknown objects. In *ICML Wksp on Statistical Relational Learning*, 2004.

[8] B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. BLOG: First-order probabilistic models with unknown objects. Technical report, UC Berkeley, 2005.

[9] H. Pasula. *Identity Uncertainty*. PhD thesis, UC Berkeley, 2003.

[10] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS 15*. MIT Press, Cambridge, MA, 2003.

[11] A. Pfeffer and D. Koller. Semantics and inference for recursive probability models. In *Proc. 17th AAAI*, 2000.

[12] D. Poole and N. L. Zhang. Exploiting contextual independence in probabilistic inference. *JAIR*, 18:263–313, 2003.

[13] S. Russell. Identity uncertainty. In *Proc. 9th Int'l Fuzzy Systems Assoc. World Congress*, 2001.

[14] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Morgan Kaufmann, 2nd edition, 2003.

[15] R. D. Shachter. Evaluating influence diagrams. *Op. Res.*, 34:871–882, 1986.