

Logical Compilation of Bayesian Networks with Discrete Variables

Michael Wachter¹ and Rolf Haenni^{1,2}

¹ University of Bern, Switzerland
{wachter,haenni}@iam.unibe.ch

² Bern University of Applied Sciences, Switzerland
rolf.haenni@bfh.ch

Abstract. This paper presents a new approach to inference in Bayesian networks. The principal idea is to encode the network by logical sentences and to compile the resulting encoding into an appropriate form. From there, all possible queries are answerable in linear time relative to the size of the logical form. Therefore, our approach is a potential solution for real-time applications of probabilistic inference with limited computational resources. The underlying idea is similar to both the differential and the weighted model counting approach to inference in Bayesian networks, but at the core of the proposed encoding we avoid the transformation from discrete to Boolean variables. This alternative encoding enables a more natural solution.

1 Introduction

As Bayesian networks (BN) are more and more applied to complex real-world applications, the development of fast and flexible inference methods becomes increasingly important. In the last decades, researchers have developed various kinds of exact and approximate inference algorithms, each of them with corresponding advantages and disadvantages. Some methods are particularly designed for real-time inference with limited computational resources such as time or memory. See [1] for a comprehensive and compact survey.

Two particular inference methods are the weighted model counting [2] and the differential approach [3, 4]. The former suggests to view a BN as a CNF model counting problem. It encodes the given BN as a CNF, and employs techniques used in the state-of-the-art SAT and model counting engines to solve the problem. The differential approach suggests to view a BN as a *multi-linear function* (MLF), the so-called *network polynomial*, from which answers to probabilistic queries are retrieved by differentiating the polynomial. Relative to the given BN, the network polynomial is exponential in size, but it is possible to efficiently encode it by a CNF of linear size. As suggested in [5], this CNF is then compiled into a *decomposable negation normal form* (DNNF) with the additional properties of *smoothness* and *determinism* [6]. The resulting sd-DNNF is an intermediate step, from which an arithmetic circuit is extracted, whose size is not necessarily exponential relative to the original BN. This arithmetic circuit

is guaranteed to compute the original network polynomial, and can therefore be used to obtain all necessary partial derivatives in time (and space) linear to its size. In its essence, the aim of the whole procedure is to generate a preferably optimal factoring of the network polynomial.

Such a logical approach is beneficial in many ways. The most important advantage is the ability to encode *context-specific independences* or other local regularities in the given conditional probability tables [4, 7]. This inherently includes appropriate solutions for the particular type of CPT obtained from noisy-OR and noisy-AND nodes or from purely logical relations. In comparison with classical inference methods such as join-tree propagation or message-passing, which do not directly exploit such local structures, there has been reports of tremendous improvements in both compile time and online inference [8, 9]. Another advantage is the ability to efficiently update numerical computations with minimal computational overhead. This is a key prerequisite for experimental sensitivity analyses.

1.1 Overview of the Method

In [10], we showed how to use the CNF encoding from [2] as a starting point for a compilation in the sense of [3, 5], but our analysis was restricted to Boolean variables only. In this paper, we will break out of the Boolean case and discuss the compilation of BNs with (finite) discrete variables. For this, a discrete variable $\Theta_{X|\mathbf{y}}$ is attributed to each row $P(X|\mathbf{y})$ in the CPT of a network variable X with a finite set of states Ω_X , e.g. $\Omega_X = \{high, medium, low\}$. As a result, the generated logical representation ψ consists of two types of variables, the ones linked to the CPT entries and the network variables. The corresponding sets of variables are denoted by Θ and Δ , respectively.

In order to use the logical representation ψ to compute the posterior probability $P(\mathbf{q}|\mathbf{e}) = P(\mathbf{q}, \mathbf{e})/P(\mathbf{e})$ of a query event $\mathbf{q} \in \Omega_{\mathbf{Q}}$, given the evidence $\mathbf{e} \in \Omega_{\mathbf{E}}$, where $\mathbf{q} = q_1 \cdots q_r$, $\Omega_{\mathbf{Q}} = \Omega_{Q_1} \times \cdots \times \Omega_{Q_r}$, $\mathbf{e} = e_1 \cdots e_s$, and $\Omega_{\mathbf{E}} = \Omega_{E_1} \times \cdots \times \Omega_{E_s}$, it is sufficient to look at the simpler problem of computing prior probabilities $P(\mathbf{x})$ of arbitrary conjunctions $\mathbf{x} = x_1 \cdots x_t \in \Omega_{\mathbf{X}} = \Omega_{X_1} \times \cdots \times \Omega_{X_t}$ in order to obtain corresponding numerators $P(\mathbf{q}, \mathbf{e})$ and denominators $P(\mathbf{e})$. Our solution for this consists of the following three steps:

1. Condition ψ on $\mathbf{x} \in \Omega_{\mathbf{X}}$ to obtain $\psi|\mathbf{x}$.
2. Eliminate (forget) from $\psi|\mathbf{x}$ the variables Δ . The resulting logical representation of $[\psi|\mathbf{x}]^{-\Delta}$ consists of variables from Θ only.
3. Compute the probability of the event represented by $[\psi|\mathbf{x}]^{-\Delta}$ to obtain $P(\mathbf{x}) = P([\psi|\mathbf{x}]^{-\Delta})$. For this, we assume that the variables $\Theta_{X|\mathbf{y}} \in \Theta$ are probabilistically independent and that $P(\theta_{x|\mathbf{y}}) = P(x|\mathbf{y})$ are the respective marginal probabilities for $\theta_{x|\mathbf{y}} \in \Theta_{X|\mathbf{y}}$ and $x \in \Omega_X$.

For the choice of an appropriate target compilation language for ψ , it is thus necessary to select a language that supports two transformations (conditioning and forgetting) and one query (probability computation) in polynomial time. At first

sight, just by looking at the results given in [11], it seems that no such language exists. However, as we will see in this paper, we can exploit the fact that the variables in Δ satisfy a certain property w.r.t. ψ . The particular form of forgetting such *deterministic* variables is called *deterministic forgetting*, and we shall see that it is (at least) supported by two languages. However, since probability computations are only supported by one of them our search for an appropriate target compilation language for Bayesian networks thus leads to *multi-state directed acyclic graphs* (MDAG), satisfying the properties of *decomposability*, *determinism*, and *no-negations*, denoted as *cdn-MDAG*. This language is the only representation language that supports all necessary operations of the above procedure in polynomial time. The suggested use of *cdn-MDAGs* has an obvious advantage: There is no need to replace discrete variables with $\ell > 2$ states by $\ell - 1$ (or ℓ) Boolean variables. Both the weighted model counting and the differential approach are based on this replacement.

1.2 Contribution and Outline

The conclusion that *cdn-MDAGs* (the multi-state version of *d-DNNFs*) should be used as target compilation language for Bayesian networks confirms Darwiche’s precursory work in [5], but it also shows that Darwiche’s additional requirement of smoothness is dispensable. While the actual reasons for this conclusion and the exact role of smoothness remain rather nebulous in [5], a precise and conclusive explanation in terms of the (extended) knowledge compilation map is given in this paper.

Another contribution of this paper is the proposal of an alternative encoding, which finally enables a more direct computational procedure in terms of a few basic operations of the knowledge compilation map. In our opinion, this is a significant simplification over Darwiche’s original method of viewing posterior probabilities as partial derivatives of multi-linear functions, from which the rather cumbersome process of transforming the CNF encoding via a smooth *d-DNNF* to an arithmetic circuit (with all negative literals set to 1) results. In the light of this paper, some steps of this process appear as an unnecessary detour, e.g., transforming multi-state variables into Boolean variables within the CNF encoding. In a nutshell, we believe that the method of this paper is an important contribution to the area of compiling Bayesian networks, mainly as a significant advancement in terms of clarity and simplicity.

The structure of this paper is as follows. Section 2 provides a short summary of possible representations of *Cartesian indicator functions* (CIF) and the corresponding knowledge compilation map. We introduce also the concepts of deterministic variables and deterministic forgetting, and extend the knowledge compilation map accordingly. The topic of Section 3 is the logical representation and evaluation of Bayesian networks. This part includes the main theorems of the paper. Section 4 concludes the paper.

2 Representing Cartesian Indicator Functions

Let $\mathbf{V} = \{V_1, \dots, V_v\}$ be a set of v variables and suppose that Ω_{V_i} denotes the finite set of states of V_i . A finite indicator function f is defined by $f : \Omega_{\mathbf{V}} \rightarrow \mathbb{B}$, where $\Omega_{\mathbf{V}} = \Omega_{V_1} \times \dots \times \Omega_{V_v}$ and $\mathbb{B} = \{0, 1\}$. To emphasize the fact that f is a mapping from the Cartesian product $\Omega_{V_1} \times \dots \times \Omega_{V_v}$ to \mathbb{B} , f is called a *Cartesian indicator function* (CIF). The so-called *satisfying set* $S_f = \{\mathbf{x} \in \Omega_{\mathbf{V}} : f(\mathbf{x}) = 1\} = f^{-1}(1)$ of f is the set of v -dimensional vectors $\mathbf{x} \in \Omega_{\mathbf{V}}$ for which f evaluates to 1. Composed CIFs will be specified by using the logical connectors $\wedge, \vee, \neg, \rightarrow$, and \leftrightarrow in their usual interpretation. Special cases of finite CIFs are Boolean functions (BF), where $\Omega_{V_i} = \mathbb{B}$, and therefore $\Omega_{\mathbf{V}} = \mathbb{B}^v$.

2.1 Representation Languages

To start with the least restrictive view w.r.t. possible representation languages, consider the concept of a *multi-state* DAG (or MDAG for short). According to [11], MDAGs are rooted, directed, acyclic graphs, in which each leaf node is represented by \square and labeled with \top (true), \perp (false), or $X=x$, where $X \in \mathbf{V}$ is a variable, and $x \in \Omega_X$ is one of its states. Each non-leaf node is represented by Δ (logical and), ∇ (logical or), or \diamond (logical not). The set of all possible MDAGs of \mathbf{V} is called *language* and denoted by $\text{MDAG}_{\mathbf{V}}$ or simply MDAG . In a MDAG, each node α represents a finite CIF f_{α} by

$$f_{\alpha} = \begin{cases} \bigwedge_{i=1}^t f_{\beta_i}, & \text{if } \alpha \text{ is an } \Delta\text{-node with children } \beta_1, \dots, \beta_t, \\ \bigvee_{i=1}^t f_{\beta_i}, & \text{if } \alpha \text{ is an } \nabla\text{-node with children } \beta_1, \dots, \beta_t, \\ \neg f_{\psi}, & \text{if } \alpha \text{ is a } \diamond\text{-node with the child } \psi, \\ 1, & \text{if } \alpha \text{ is a } \square\text{-node labeled with } \top, \\ 0, & \text{if } \alpha \text{ is a } \square\text{-node labeled with } \perp, \\ f_{X=x}, & \text{if } \alpha \text{ is a } \square\text{-node labeled with } X=x, \end{cases}$$

where $f_{X=x}(\mathbf{x})$ with $\mathbf{x} \in \Omega_{\mathbf{V}}$ is defined by

$$f_{X=x}(\mathbf{x}) = \begin{cases} 1, & \text{if } x \text{ is the corresponding value of } X \text{ in } \mathbf{x}, \\ 0, & \text{otherwise.} \end{cases}$$

The MDAG depicted in Fig. 1 represents the finite CIF $f = ([Y=y_1] \wedge [X=x_1]) \vee ([Y=y_2] \wedge \neg[X=x_2]) \vee ([X=x_2] \wedge [Y=y_3])$.

Our convention is to denote MDAGs by lower-case Greek letters such as φ, ψ , or the like. Two MDAGs $\varphi, \psi \in \text{MDAG}$ are *equivalent*, denoted by $\varphi \equiv \psi$, iff $f_{\varphi} = f_{\psi}$. Furthermore, φ *entails* ψ , denoted by $\varphi \models \psi$, iff $f_{\varphi}(\mathbf{x}) \leq f_{\psi}(\mathbf{x})$ for all $\mathbf{x} \in \Omega$. The set of variables included in $\varphi \in \text{MDAG}$ is denoted by $\text{vars}(\varphi) \subseteq \mathbf{V}$. The number of edges of φ is called its *size* and is denoted by $|\varphi|$. MDAGs may satisfy various properties [11], but in the context of this paper, only three of them are relevant:

- *Decomposability* (c): the sets of variables of the children of each Δ -node α in φ are pairwise disjoint (i.e. if β_1, \dots, β_n are the children of α , then $\text{vars}(\beta_i) \cap \text{vars}(\beta_j) = \emptyset$ for all $i \neq j$);

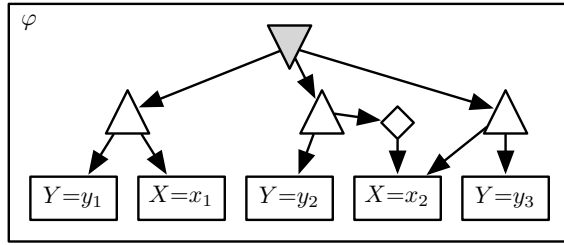


Fig. 1. The finite CIF f represented as the MDAG φ .

- *Determinism* (d): the children of each \forall -node α in φ are pairwise logically contradictory (i.e. if β_1, \dots, β_n are the children of α , then $\beta_i \wedge \beta_j \equiv \perp$ for all $i \neq j$);
- *No-Negation* (n):³ φ does not contain any \diamond -node.

A decomposable and deterministic MDAG is called **cd-MDAG**, and **cd-MDAG** refers to the corresponding language, a sub-language of **MDAG**. The example shown in Fig. 1 is a **cd-MDAG**.

Another important sub-language is **cn-MDAG**. It refers to the sub-languages of **MDAG** where decomposability and no-negation are satisfied. **cdn-MDAG** is the sub-language of **cn-MDAG**, where determinism is satisfied in addition to decomposability and no-negation. This languages includes Darwiche’s **d-DNNF** language as a special case, when all variables are Boolean. Other sub-languages are obtained from considering further properties, e.g. **OMDD** (ordered multivalued decision diagram) is the sub-language of **cdn-MDAG** satisfying *decision*, *read-once*, and *ordering*. For a more comprehensive overview and a detailed discussion we refer to [11].

2.2 Succinctness, Queries, and Transformations

A language L_1 is *equally or more succinct* than another language L_2 , $L_1 \preceq L_2$, if any sentence $\alpha_2 \in L_2$ has an equivalent sentence $\alpha_1 \in L_1$ whose size is polynomial in the size of α_2 . A language L_1 is *strictly more succinct* than another language L_2 , $L_1 \prec L_2$, iff $L_1 \preceq L_2$ and $L_2 \not\preceq L_1$. With respect to the above-mentioned languages, we have the following proven relationships [11]:

$$\text{MDAG} \prec \left\{ \begin{array}{l} \text{cn-MDAG} \prec \\ \text{cd-MDAG} \preceq \end{array} \right\} \text{cdn-MDAG} \prec \text{OMDD}.$$

It is still unknown whether **cd-MDAG** is strictly more succinct than **cdn-MDAG** or not.

Queries are operations that return information about a finite CIF without changing its MDAG representation. The most important queries are *consistency* (**CO**), *validity* (**VA**), *clause entailment* (**CE**), *term implication* (**IM**), *senten-*

³ *No-negation* corresponds to *simple-negation* in [12, 13]

tial entailment (SE), equivalence (EQ), model counting (CT), probabilistic equivalence (PEQ), and probability computation (PR).

Finally, a *transformation* is an operation that returns a MDAG representing a modified finite CIF. The new MDAG is supposed to satisfy the same properties as the language in use. The most important transformations are (*term conditioning* (TC), *forgetting* (FO), *singleton forgetting* (SFO), *general/binary conjunction* (AND/AND₂), *general/binary disjunction* (OR/OR₂), and *negation* (NOT).

If a language supports a query or transformation in polynomial time with respect to the size of the involved MDAGs, we say that it *supports* this query or transformation. Table 1 shows the supported queries and transformations of the considered languages [11].

	CO/CE	VA/IM	CT/PR/PEQ	EQ	SE	TC	FO	SFO	AND	AND ₂	OR	OR ₂	NOT
MDAG	◦	◦	◦	◦	◦	√	◦	√	√	√	√	√	√
cn-MDAG	√	◦	◦	◦	◦	√	√	√	◦	◦	√	√	◦
cd-MDAG	√	√	√	?	◦	√	◦	◦	◦	◦	◦	◦	√
cdn-MDAG	√	√	√	?	◦	√	◦	◦	◦	◦	◦	◦	?
OMDD	√	√	√	√	◦	√	•	√	•	◦	•	◦	√

Table 1. Sub-languages of the MDAG language and their supported queries and transformations. √ means "supports", • means "does not support", ◦ means "does not support unless $P = NP$ ", and ? means "unknown".

2.3 Deterministic Variables

It is interesting to see in Table 1 that forgetting is supported by cn-MDAG but not by cdn-MDAG or cd-MDAG. This is a consequence of the fact that forgetting does not preserve determinism in general. Let us now have a look at the particular case of variables which preserve determinism while being forgotten.

Definition 1. For $\varphi \in \text{MDAG}$, the variable $X \in \mathbf{V}$ is called *deterministic w.r.t.* φ , denoted by $X \parallel \varphi$, iff $[\varphi|x] \wedge [\varphi|x'] \equiv \perp$ for all states $x, x' \in \Omega_X$, $x \neq x'$.

This corresponds to [14], i.e. the value of the variable X is *determined* by the variables $\mathbf{V} \setminus \{X\}$ in φ . The process of forgetting deterministic variables will be discussed in the next subsection. Before, let's have a look at some basic properties of deterministic variables.

Theorem 1. $X \parallel \varphi$ implies $X \parallel \psi$ for all $\psi \models \varphi$.

Theorem 2. $X \notin \text{vars}(\varphi)$ implies $X \parallel X \leftrightarrow \varphi$.

The proofs are analog to the proofs of the corresponding theorems in [10]. An immediate consequence is the following corollary, which is necessary to prove one of the main theorems of Section 3.

Corollary 1. $X \notin \text{vars}(\varphi)$ implies $X \parallel (X \leftrightarrow \varphi) \wedge \psi$.

For the forgetting of more than one variable, it is useful to generalize the definition of a single deterministic variable to sets of deterministic variables.

Definition 2. For $\varphi \in \text{MDAG}$, the set of variables $\mathbf{X} = \{X_1, \dots, X_n\} \subseteq \mathbf{V}$ is called deterministic w.r.t φ , denoted by $\mathbf{X} \parallel \varphi$ or simply $X_1, \dots, X_n \parallel \varphi$, iff $[\varphi|\mathbf{x}] \wedge [\varphi|\mathbf{x}'] \equiv \perp$ for all instantiations $\mathbf{x}, \mathbf{x}' \in \Omega_{\mathbf{X}}$, $\mathbf{x} \neq \mathbf{x}'$.

Note that $X, Y \parallel \varphi$ implies $X \parallel \varphi$ and $Y \parallel \varphi$, while the converse is not always true.

2.4 Deterministic Forgetting

Let $\mathbf{W} \subseteq \mathbf{V}$ be a subset of variables, $X \in \mathbf{V}$ a single variable, and φ an arbitrary MDAG. Forgetting the variables \mathbf{W} from φ generates a new MDAG $\varphi^{-\mathbf{W}}$, in which the variables from \mathbf{W} are no longer included, and such that its satisfying set $S_{\varphi-\mathbf{w}}$ is the projection of S_{φ} to the restricted set of variables $\mathbf{V} \setminus \mathbf{W}$. In the literature, forgetting was originally called *elimination of middle terms* [15], but it is also common to call it *projection*, *variable elimination*, or *marginalization* [16]. There is also a one-to-one analogy to the elimination of existential quantifiers in *quantified (Boolean) formulas* [17], as discussed below.

Singleton forgetting is forgetting with $\mathbf{W} = \{X\}$. A general and simple way to realize singleton forgetting is by constructing a MDAG of the form

$$\varphi^{-X} = \bigvee_{x \in \Omega_X} [\varphi|x].$$

Note that if X is Boolean, φ^{-X} is logically equivalent to the quantified Boolean formula $(\exists x)\varphi$. It is easy to see that singleton forgetting preserves the properties of simple-negation and decomposability (if present), while determinism is not preserved (the children of the new ∇ -node are not necessarily logically contradictory). This is the reason why singleton forgetting is only supported by MDAG and **cn**-MDAG, but not by **cd**-MDAG or **cdn**-MDAG (see Table 1).

Forgetting multiple variable is usually realized as a sequence of singleton forgetting. In general, this may result in an exponential blow-up of the MDAG size, but the decomposability of **cn**-MDAG allows to keep this blow-up under control. This is the reason why **cn**-MDAG is the only language to support forgetting in general. For the details of a corresponding algorithm (for Boolean variables), we refer to [18].

Now let's turn our attention to the special case of forgetting deterministic variables. One way to look at it is to define two additional transformations called *deterministic forgetting* (**FO_d**) and *deterministic singleton forgetting* (**SFO_d**). They correspond to **FO** and **SFO**, respectively, but are only applicable to deterministic variables.

For $\mathbf{X} \parallel \varphi$, the children of the new ∇ -node of $\bigvee_{\mathbf{x} \in \Omega_{\mathbf{X}}} [\varphi|\mathbf{x}]$ are logically contradictory by definition. In other words, forgetting deterministic variables preserves

determinism. Thus, we can use the forgetting algorithm of **cn-MDAG** for forgetting deterministic variables in the context of **cdn-MDAG**. As a consequence, **SFO_d** and **FO_d** are both supported by **cdn-MDAG**, as stated in the following theorem.

Theorem 3.

- a) MDAG supports **SFO_d**, but it does not support **FO_d** unless $P = NP$.
- b) **cn-MDAG** and **cdn-MDAG** support **FO_d** and **SFO_d**.
- c) **cd-MDAG** and **OMDD** support **SFO_d**.

The proof is analog to the one given in [10]. Whether **cd-MDAG** and **OMDD** support **FO_d** is an open question.

3 Compiling Bayesian Networks

The goal of this section is to show that the probability distribution induced by a Bayesian network can be represented by a so-called *multi-state CNF* (MCNF) [11] (1st subsection) and that the **cdn-MDAG** compilation of this MCNF can be used to efficiently compute arbitrary posterior probabilities (2nd subsection). The proposed MCNF representation is similar but not equivalent to the one proposed by Darwiche in [5], for details consider [10]. However, if all variables of the network are Boolean, the MCNF is equivalent to the CNF proposed in [2].

A *Bayesian network* (BN) is a compact graphical model of a complex probability distribution over a set of variables $\Delta = \{X_1, \dots, X_n\}$ [19]. It consists of two parts: a DAG representing the direct influences among the variables, and a set of conditional probability tables (CPT) quantifying the strengths of these influences. The whole BN represents the exponentially sized *joint probability distribution* over its variables in a compact manner by

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{parents}(X_i)),$$

where $\text{parents}(X_i)$ denotes the parents of node X_i in the DAG. Figure 2 depicts a small BN with three (Boolean) variables X , Y , and Z .

3.1 Logical Representation

Consider a variable $X \in \Delta$ with $\text{parents}(X) = \{Y_1, \dots, Y_u\} = \mathbf{Y}$, $\Omega_X = \{x_1, \dots, x_t\}$, and the corresponding CPT. Since X has u parents, the CPT will have $|\Omega_{\mathbf{Y}}| \geq 2^u$ rows. The CPT rows contain the conditional probability distributions of the states of X given the corresponding instantiation $\mathbf{y} \in \Omega_{\mathbf{Y}}$ of \mathbf{Y} . For each probability distribution $P(X|\mathbf{y})$, we introduce an auxiliary variable $\Theta_{X|\mathbf{y}}$ with $\Omega_{\Theta_{X|\mathbf{y}}} = \{\theta_{x_1|\mathbf{y}}, \dots, \theta_{x_t|\mathbf{y}}\}$. Assuming that the variables $\Theta_{X|\mathbf{y}}$ represent probabilistically independent events, we define their respective marginal probabilities by $P(\theta_{x|\mathbf{y}}) = P(x|\mathbf{y})$, as shown in Fig. 2.

To see how the proposed logical representation of the BN works, take a closer look at one particular instantiation \mathbf{y} of $\text{parents}(X)$. The idea is that

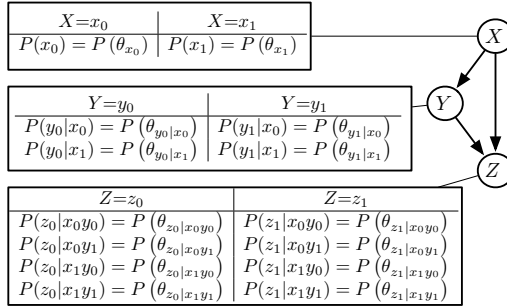


Fig. 2. Example of a Bayesian network.

if \mathbf{y} happens to be the true state of $\text{parents}(X)$, then $\Theta_{X|\mathbf{y}} = \theta_{x|\mathbf{y}}$ is supposed to logically imply $X=x$. This logical relationship between $Y_1=y_1, \dots, Y_u=y_u$, $\Theta_{X|\mathbf{y}} = \theta_{x|\mathbf{y}}$ with $\mathbf{y} = y_1 \cdots y_u \in \Omega_{\mathbf{Y}}$ and $X=x$ is expressed by the implication in the following logical expression. By taking the conjunction of all such implications over all instantiations \mathbf{y} , we obtain a logical representation ψ_X of the node X with its relationship to its parents:

$$\psi_X = \bigwedge_{\substack{\mathbf{y} \in \Omega_{\mathbf{Y}} \\ \mathbf{y} = y_1 \cdots y_u}} \left(\bigwedge_{\theta_{x|\mathbf{y}} \in \Omega_{\Theta_{X|\mathbf{y}}}} \left([Y_1=y_1] \wedge \cdots \wedge [Y_u=y_u] \wedge [\Theta_{X|\mathbf{y}} = \theta_{x|\mathbf{y}}] \rightarrow [X=x] \right) \right)$$

A logical representation ψ_{Δ} of the whole BN is the conjunction

$$\psi_{\Delta} = \bigwedge_{X \in \Delta} \psi_X$$

over all network variables $X \in \Delta$. This sentence includes two types of variables, the ones linked to the CPT entries and the network variables. The respective sets of variables are denoted by Θ and Δ , respectively.⁴ Note that ψ_X and therewith

⁴ The representation of a Bayesian network by a logical sentence ψ_{Δ} over two sets of variables Θ and Δ , together with the given marginal probabilities for the variables in Θ and the corresponding independence assumptions, puts this approach in the broader context of *probabilistic argumentation* [20, 21]. This is a theory of formal reasoning which aims at unifying the classical fields of logical and probabilistic reasoning. The principal idea is to evaluate the credibility of a hypothesis by non-additive *probabilities of provability* (or *degrees of support*). This is a natural extension of the classical concepts of probability (in probability theory) and provability (in logic) [20]. The non-additivity of this measure is an important characteristic to distinguish properly between uncertainty and ignorance, but the particularity of the model in this paper always causes the resulting probabilities of provability to degenerate into ordinary (additive) probabilities. The embedding into the theory of probabilistic argumentation has no practical significance for the method and goals of this paper, but it allows inference in Bayesian network to be seen from a totally new perspective. We expect this perspective to be useful as a starting point to study inference in Bayesian networks with missing parameters.

ψ_Δ is a MCNF, as each of its implications can be written as a clause. For the BN of Fig. 2, we have $\Theta = \{\Theta_X, \Theta_{Y|x_0}, \Theta_{Y|x_1}, \Theta_{Z|x_0y_0}, \Theta_{Z|x_0y_1}, \Theta_{Z|x_1y_0}, \Theta_{Z|x_1y_1}\}$, $\Delta = \{X, Y, Z\}$, and

$$\psi_\Delta = \bigwedge \left(\begin{array}{l} \left. \begin{array}{l} [\Theta_X = \theta_{x_0}] \rightarrow [X = x_0] \\ [\Theta_X = \theta_{x_1}] \rightarrow [X = x_1] \end{array} \right\} \text{from } \psi_X \\ \left. \begin{array}{l} [X = x_0] \wedge [\Theta_{Y|x_0} = \theta_{y_0|x_0}] \rightarrow [Y = y_0] \\ [X = x_0] \wedge [\Theta_{Y|x_0} = \theta_{y_1|x_0}] \rightarrow [Y = y_1] \\ [X = x_1] \wedge [\Theta_{Y|x_1} = \theta_{y_0|x_1}] \rightarrow [Y = y_0] \\ [X = x_1] \wedge [\Theta_{Y|x_1} = \theta_{y_1|x_1}] \rightarrow [Y = y_1] \end{array} \right\} \text{from } \psi_Y \\ \left. \begin{array}{l} [X = x_0] \wedge [Y = y_0] \wedge [\Theta_{Z|x_0y_0} = \theta_{z_0|x_0y_0}] \rightarrow [Z = z_0] \\ [X = x_0] \wedge [Y = y_0] \wedge [\Theta_{Z|x_0y_0} = \theta_{z_1|x_0y_0}] \rightarrow [Z = z_1] \\ [X = x_0] \wedge [Y = y_1] \wedge [\Theta_{Z|x_0y_1} = \theta_{z_0|x_0y_1}] \rightarrow [Z = z_0] \\ [X = x_0] \wedge [Y = y_1] \wedge [\Theta_{Z|x_0y_1} = \theta_{z_1|x_0y_1}] \rightarrow [Z = z_1] \\ [X = x_1] \wedge [Y = y_0] \wedge [\Theta_{Z|x_1y_0} = \theta_{z_0|x_1y_0}] \rightarrow [Z = z_0] \\ [X = x_1] \wedge [Y = y_0] \wedge [\Theta_{Z|x_1y_0} = \theta_{z_1|x_1y_0}] \rightarrow [Z = z_1] \\ [X = x_1] \wedge [Y = y_1] \wedge [\Theta_{Z|x_1y_1} = \theta_{z_0|x_1y_1}] \rightarrow [Z = z_0] \\ [X = x_1] \wedge [Y = y_1] \wedge [\Theta_{Z|x_1y_1} = \theta_{z_1|x_1y_1}] \rightarrow [Z = z_1] \end{array} \right\} \text{from } \psi_Z \end{array} \right).$$

3.2 Computing Posterior Probabilities

The goal of a BN is the computation of the posterior probability $P(\mathbf{q}|\mathbf{e}) = P(\mathbf{q}, \mathbf{e})/P(\mathbf{e})$ of a query event $\mathbf{q} = q_1 \cdots q_r \in \Omega_{\mathbf{Q}}$ given the observed evidence $\mathbf{e} = e_1 \cdots e_s \in \Omega_{\mathbf{E}}$. As mentioned in Section 1, it is sufficient to look at the simpler problem of computing prior probabilities $P(\mathbf{x})$ of arbitrary conjunctions $\mathbf{x} \in \Omega_{\mathbf{X}}$. The following theorem states that the essential step to solve this problem is to forget the propositions Δ from ψ_Δ (or any equivalent form of it) conditioned on \mathbf{x} .

Theorem 4. $P(\mathbf{x}) = P([\psi_\Delta|\mathbf{x}]^{-\Delta})$.

This guarantees that the computed values are correct. To ensure that this computation requires only polynomial time, we need to compile ψ_Δ into an appropriate language, one that simultaneously supports TC, FO, and PR. According to Table 1, there is no such language, but the following theorem allows us to replace FO, not supported by *cdn*-MDAG, by FO_d, supported by *cdn*-MDAG.

Theorem 5. $\Delta \parallel \psi_\Delta$.

As a consequence of this simple theorem, we arrive at the main message of this paper, namely that *cdn*-MDAG is the most suitable target compilation language for Bayesian networks, since it supports TC, FO_d, and PR, and thus allows to compute posterior probabilities in polynomial time.

For the compilation of the MCNF ψ_Δ into a *cdn*-MDAG, we can use the state-of-the-art CNF to d-DNNF or any CNF to OBDD compiler [6, 22] in the Boolean case. For the general case, we are currently working on the adaption of these algorithms.

4 Conclusion

The approach proposed in this paper extends a logical inference method for Bayesian networks with Boolean variables to Bayesian networks with multi-state variables. We expect its contribution to be theoretically and practically significant. On the theoretical side, based on an extended knowledge compilation map, the paper provides a precise explanation of why cdn-MDAGs are apparently the most suitable logical representations for Bayesian networks. This is mainly a consequence of the fact that some of the involved variables are deterministic. The paper also demonstrates how to reduce the problem of logical inference in Bayesian networks to three basic logical operations. Compared to Darwiche's differential approach, this view fits much better into the picture of the knowledge compilation map, as the reduction to these essential elements no longer requires us to talk about network polynomials, multi-linear functions, partial derivatives, arithmetic circuits, or smoothness. In this sense, we also see our paper as an attempt to clarify the theoretical mechanisms and connections behind this kind of inference algorithms and as a good example to demonstrate the usefulness of the knowledge compilation map.

On the practical side, the paper provides precise step-by-step instructions to implement a new encoding and inference method for Bayesian networks in terms of a few simple operations for cdn-MDAGs. Compared to Darwiche's differential approach, this will lead to more transparent implementations. Finally, with respect to possible applications other than Bayesian networks, other situations with deterministic variables may be detected, for which forgetting becomes tractable in the case of cdn-MDAGs.

Acknowledgment Research supported by the *Swiss National Science Foundation*, Project No. PP002-102652/1, and *The Leverhulme Trust*. Special thanks to Adnan Darwiche for his very helpful comments.

References

1. Guo, H., Hsu, W.H.: A survey of algorithms for real-time Bayesian network inference. In Darwiche, A., Friedman, N., eds.: *AAAI/KDD/UAI'02, Joint Workshop on Real-Time Decision Support and Diagnosis Systems*, Edmonton, Canada (2002)
2. Sang, T., Beame, P., Kautz, H.: Solving Bayesian networks by weighted model counting. In: *AAAI'05, 20th National Conference on Artificial Intelligence*. Volume 1., Pittsburgh, USA (2005) 475–482
3. Darwiche, A.: A differential approach to inference in Bayesian networks. *Journal of the ACM* **50**(3) (2003) 280–305
4. Chavira, M., Darwiche, A.: Compiling Bayesian networks using variable elimination. In: *IJCAI'07, 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India (2007)
5. Darwiche, A.: A logical approach to factoring belief networks. In Fensel, D., Giunchiglia, F., McGuinness, D.L., Williams, M.A., eds.: *KR'02, 8th International Conference on Principles and Knowledge Representation and Reasoning*, Toulouse, France (2002) 409–420

6. Darwiche, A.: A compiler for deterministic, decomposable negation normal form. In: AAAI'02, 18th National Conference on Artificial Intelligence, Edmonton, Canada (2002) 627–634
7. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In Horvitz, E., Jensen, F., eds.: UAI'96, 12th Conference on Uncertainty in Artificial Intelligence, Portland, USA (1996) 115–123
8. Chavira, M., Darwiche, A.: Compiling Bayesian networks with local structure. In: IJCAI'05, 19th International Joint Conference on Artificial Intelligence, Edinburgh, U.K. (2005)
9. Chavira, M., Darwiche, A., Jaeger, M.: Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning* **42**(1–2) (2006) 4–20
10. Wachter, M., Haenni, R.: Logical compilation of Bayesian networks. Technical Report iam-06-006, University of Bern, Switzerland (2006)
11. Wachter, M., Haenni, R.: Multi-state directed acyclic graphs. In Kobti, Z., Wu, D., eds.: CanAI'07, 20th Canadian Conference on Artificial Intelligence. LNAI 4509, Montréal, Canada (2007) 464–475
12. Darwiche, A., Marquis, P.: A knowledge compilation map. *Journal of Artificial Intelligence Research* **17** (2002) 229–264
13. Wachter, M., Haenni, R.: Propositional DAGs: a new graph-based language for representing Boolean functions. In Doherty, P., Mylopoulos, J., Welty, C., eds.: KR'06, 10th International Conference on Principles of Knowledge Representation and Reasoning, Lake District, U.K., AAAI Press (2006) 277–285
14. Palacios, H., Bonet, B., Darwiche, A., Geffner, H.: Pruning conformant plans by counting models on compiled d-DNNF representations. In: ICAPS'05, 15th International Conference on Planning and Scheduling, Monterey, USA (2005) 141–150
15. Boole, G.: *The Laws of Thought*. Walton and Maberley, London (1854)
16. Kohlas, J.: *Information Algebras: Generic Structures for Inference*. Springer, London (2003)
17. Davis, S., Putnam, M.: A computing procedure for quantification theory. *Journal of the ACM* **7**(3) (1960) 201–215
18. Darwiche, A.: Decomposable negation normal form. *Journal of the ACM* **48**(4) (2001) 608–647
19. Pearl, J.: *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, USA (1988)
20. Haenni, R.: Towards a unifying theory of logical and probabilistic reasoning. In Cozman, F.B., Nau, R., Seidenfeld, T., eds.: ISIPTA'05, 4th International Symposium on Imprecise Probabilities and Their Applications, Pittsburgh, USA (2005) 193–202
21. Haenni, R., Kohlas, J., Lehmann, N.: Probabilistic argumentation systems. In Gabbay, D.M., Smets, P., eds.: *Handbook of Defeasible Reasoning and Uncertainty Management Systems*. Volume 5: Algorithms for Uncertainty and Defeasible Reasoning. Kluwer Academic Publishers, Dordrecht, Netherlands (2000) 221–288
22. Darwiche, A.: New advances in compiling CNF to decomposable negational normal form. In: ECAI'04, 16th European Conference on Artificial Intelligence, Valencia, Spain (2004)