JMLR: Workshop and Conference Proceedings 13: 161-176
2nd Asian Conference on Machine Learning (ACML2010), Tokyo, Japan, Nov. 8–10, 2010.

# An EM algorithm on BDDs with order encoding for logic-based probabilistic models

**Masakazu Ishihata** ISHIHATA@MI.CS.TITECH.AC.JP
**Yoshitaka Kameya** KAMEYA@MI.CS.TITECH.AC.JP
**Taisuke Sato** SATO@MI.CS.TITECH.AC.JP
*Graduate School of Information Science and Engineering, Tokyo Institute of Technology*
*2-12-1, O-okayama, Meguro-ku, Tokyo, 152-8552, Japan*

**Shin-ichi Minato** MINATO@IST.HOKUDAI.AC.JP
*Graduate School of Information Science and Technology, Hokkaido University*
*North14 West9, Sapporo, 060-0814, Japan*

**Editor:** Masashi Sugiyama and Qiang Yang

## Abstract

Logic-based probabilistic models (LBPMs) enable us to handle problems with uncertainty succinctly thanks to the expressive power of logic. However, most of LBPMs have restrictions to realize efficient probability computation and learning. We propose an EM algorithm working on BDDs with order encoding for LBPMs. A notable advantage of our algorithm over existing approaches is that it copes with multi-valued random variables without restrictions. The complexity of our algorithm is proportional to the size of a BDD representing observations. We utilize our algorithm to make a diagnoses of a logic circuit which contains stochastic error gates and show that restrictions of existing approaches can be eliminated by our algorithm.

**Keywords:** binary decision diagrams, EM algorithm, order encoding, propositionalized probability computation

## 1. Introduction

*Logic-based probabilistic models* (LBPMs) (Getoor and Taskar (2007); De Raedt et al. (2008)) have been developed for over the past two decades as a way to combine probabilistic models and first-order logic. Probabilistic models can handle uncertainty whereas first-order logic can represent our knowledge efficiently. Thanks to the expressive power of first-order logic, they can succinctly deal with various problems in the real world. For instance, Inoue et al. (2009) proposed an inference system using LBPMs. Their system derives hypotheses about metabolic pathways from biological data and background knowledge described in first-order logic and evaluates them using probabilities learned from the data.

In general, an LBPM is defined by a first-order language with a distribution over an infinite number of possible worlds. Even if we assume finite possible worlds, their size is exponential in that of ground atoms (random variables). Consequently, probability computation and learning in LBPMs require exponential time. To realize efficient probability computation and/or learning, most of LBPMs put some restrictions. For example, *prob-*

*abilistic Horn abduction* (PHA) (Poole (1993)) assumes the bodies of Horn clauses which have the same head are mutually exclusive. In *stochastic logic programs* (SLPs) (Muggleton (1996)), clauses which define the same predicate must be probabilistically exclusive each other. *PRISM* (Sato and Kameya (2001)) also assumes that the disjuncts in formulas are probabilistically exclusive. Introducing such exclusiveness restrictions surely makes probability computation simple and efficient, but, it prevents us from enjoying the full expressive power of first-order logic.

To relax the exclusiveness restrictions, BDD-based probability computation and learning have been proposed (De Raedt et al. (2007); Ishihata et al. (2008); Gutmann et al. (2010)). A *binary decision diagram* (BDD) (Akers (1978)) is a rooted directed acyclic graph that gives a compact representations of a boolean function. Since probability computation and learning in LBPMs are based on propositional logic, compressing boolean (propositional) functions by BDDs accelerates them even if we do not make assumptions like the above. De Raedt et al. (2007) proposed *ProbLog* which is a probabilistic extension of Prolog and its probability computation is based on BDDs. Ishihata et al. (2008) proposed the *BDD-EM algorithm* which is a probability learning algorithm working on BDDs. Gutmann et al. (2010) also proposed a BDD-based learning algorithm for ProbLog called the *CoPrEM algorithm.* However, both learning algorithms still have a serious problem that they cannot deal efficiently with multi-valued random variables (a multi-valued random variable is a random variable which has a discrete finite domain like dice). This problem is caused by the following two reasons. One is that they assume a target distribution to be learned is a product of Bernoulli distributions over boolean variables. The other is that BDDs expressing multi-valued variables tend to be huge since its compression rules are unfit to multi-valued variables.

In this paper, we propose a new BDD-based learning algorithm which can deal with multi-valued random variables. We first introduce a propositional logic-based probabilistic model $\mathcal{M}_{\boldsymbol{A}}$ which is a pair of a set of boolean variables $\boldsymbol{A}$ and a joint distribution $P_{\boldsymbol{A}}$ over $\boldsymbol{A}$. To handle multi-valued random variables, we define $P_{\boldsymbol{A}}$ based on a product of *categorical distributions.* A Bernoulli distribution is a discrete probability distribution of a boolean random variable whereas a categorical distribution is that of a multi-valued random variable. To cope with multi-valued variables by BDDs, we introduce *order encoding* (Tamura et al. (2006)) which is a way to represent multi-valued variables by boolean variables. The combination of BDDs and order encoding can compress boolean formulas well even if they contain multi-valued variables. We propose the *BO-EM algorithm* which is a probability learning algorithm for $\mathcal{M}_{\boldsymbol{A}}$ using BDDs and order encoding. BO-EM learns parameters of $\mathcal{M}_{\boldsymbol{A}}$, *i.e.* those of categorical distributions, efficiently thanks to the compression power of the combination of BDDs and order encoding. As a result, it can eliminate the exclusive restrictions in the models that contain multi-valued variables.

The rest of this paper is organized as follows. We first formulate our problem in Section 2. In Section 3, we briefly review BDDs and order encoding and then propose the BO-EM algorithm. Experimental results are reported in Section 4. Finally, we conclude and mention the related work in Section 6.

## 2. Problem settings

Suppose we observe various events in the real world. Probability is useful to express uncertainty in them whereas logic is useful to describe them. We here introduce a propositional logic-based probabilistic model $\mathcal{M}_{\boldsymbol{A}} \equiv \langle \boldsymbol{A}, P_{\boldsymbol{A}} \rangle$, where $\boldsymbol{A}$ is a set of boolean random variables and $P_{\boldsymbol{A}}$ is a joint distribution over $\boldsymbol{A}$. Probabilistic events are described by boolean formulas of $\boldsymbol{A}$ and their probabilities are defined by $P_{\boldsymbol{A}}$. To deal with multi-valued random variables by $\mathcal{M}_{\boldsymbol{A}}$, we define $P_{\boldsymbol{A}}$ as a product of categorical distributions. We next formalize the parameter estimation problem of $\mathcal{M}_{\boldsymbol{A}}$ and introduce the EM algorithm to solve it.

### 2.1 Propositional logic-based probabilistic models

We here define a $\mathcal{M}_{\boldsymbol{A}}$ representing a joint distribution over a set of multi-valued random variables. Suppose $\boldsymbol{X}$ is a set $\{X_1, \ldots, X_N\}$, where $X_i$ is a multi-valued random variable following a categorical distribution over a finite discrete domain $\mathcal{D}_{X_i} = \{v_i^{(j)} \mid 1 \le j \le N_{X_i}\}$. Let $\boldsymbol{x}$ be a sequence $\{x_i\}_i$, where $x_i \in \mathcal{D}_{X_i}$ is the assignment of $X_i$. We assume variables in $\boldsymbol{X}$ are mutually independent and also assume some of them are independent and identically-distributed (*i.i.d.* ), *i.e.* they have a common domain and a common distribution. Let $\Pi \equiv \{\pi_i \mid 1 \le i \le M, \pi_i \subseteq \boldsymbol{X}\}$ be a partition of $\boldsymbol{X}$ such that variables in the same subset are *i.i.d.* each other. Thus, $v_i^{(j)} = v_{i'}^{(j)}$ and $P(X_i = v_i^{(j)}) = P(X_{i'} = v_{i'}^{(j)})$ hold for all $j$ if $X_i$ and $X_{i'}$ are in the same subset. The partition $\Pi$ can be expressed as a mapping $\mu : \{1, \ldots, N\} \to \{1, \ldots, M\}$ such that $X_{i'} \in \pi_i \Leftrightarrow \mu(i') = i$. Let $\mathcal{D}_i$ be the common domain of $X_{i'} \in \pi_i$ and $N_i$ be the size of $\mathcal{D}_i$. To define $P_{\boldsymbol{X}}$ which is a joint distribution over $\boldsymbol{X}$, we introduce *parameters* $\boldsymbol{\theta} \equiv \{\theta_{i,j} \mid 1 \le i \le M, 1 \le j \le N_i\}$ satisfying $0 \le \theta_{i,j} \le 1$ and $\sum_{j=1}^{N_i} \theta_{i,j} = 1$. A probability of $X_i = v_i^{(j)}$ is denoted by $\theta_{\mu(i),j}$ and $P_{\boldsymbol{X}}$ is defined as follows:

$$P_{\boldsymbol{X}}(\boldsymbol{x}; \boldsymbol{\theta}) \equiv \prod_{i=1}^{N} \prod_{j=1}^{N_{X_i}} \theta_{\mu(i),j}^{\boldsymbol{x}_{(i,j)}} = \prod_{i=1}^{M} \prod_{j=1}^{N_i} \theta_{i,j}^{\sigma_{i,j}(\boldsymbol{x})},$$

where $P_{\boldsymbol{X}}(\boldsymbol{x}; \boldsymbol{\theta})$ is a shorthand for $P_{\boldsymbol{X}}(\boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta})$ and $\boldsymbol{x}_{(i,j)}$ and $\sigma_{i,j}(\boldsymbol{x})$ are defined by

$$\boldsymbol{x}_{(i,j)} \equiv \begin{cases} 1 & x_i = v_i^{(j)} \\ 0 & \textit{otherwise} \end{cases}, \qquad \sigma_{i,j}(\boldsymbol{x}) \equiv |\{i' \mid \mu(i') = i, \boldsymbol{x}_{(i',j)} = 1\}|.$$

There are many ways to represent the assignment $\boldsymbol{X} = \boldsymbol{x}$ by boolean variables and the simplest one is *direct encoding* which represents an $N$-valued variable by $N$ boolean variables. Let $\boldsymbol{A} \equiv \{A_{i,j} \mid 1 \le i \le N, 1 \le j \le N_{X_i}\}$, where $A_{i,j}$ is a boolean variable corresponding to a proposition "$X_i = v_i^{(j)}$", and also let $\boldsymbol{a} \equiv \{a_{i,j}\}_{i,j}$, where $a_{i,j}$ is the assignment of $A_{i,j}$. Direct encoding gives a mapping $d : \Omega_{\boldsymbol{X}} \to \Omega_{\boldsymbol{A}}$, where $\Omega_{\boldsymbol{X}}$ and $\Omega_{\boldsymbol{A}}$ are the set of all possible assignments of $\boldsymbol{X}$ and $\boldsymbol{A}$, respectively. The mapping $d$ is defined by

$$d(\boldsymbol{x}) = \boldsymbol{a} \iff \forall i, j \left(\boldsymbol{a}_{(i,j)} = \boldsymbol{x}_{(i,j)}\right),$$

where $\boldsymbol{a}_{(i,j)}$ is the assignment of $A_{i,j}$ in $\boldsymbol{a}$. Note that $d$ is an injection but not a surjection. Let $ran(d)$ be the range of $d$, *i.e.* $ran(d) \equiv \{d(\boldsymbol{x}) \mid \boldsymbol{x} \in \Omega_{\boldsymbol{X}}\}$. We represent $ran(d)$ by a boolean formula. A boolean formula $F$ of $\boldsymbol{A}$ is an expression of a boolean function from

$\Omega_{\boldsymbol{A}}$ to $\{0,1\}$. We use the same symbol $F$ to indicate its function. Let $C_{\boldsymbol{A}}$ be a boolean formula defined by

$$C_{\boldsymbol{A}} \equiv L_{\boldsymbol{A}} \wedge M_{\boldsymbol{A}}, \qquad L_{\boldsymbol{A}} \equiv \bigwedge_{i=1}^{N} \bigvee_{j=1}^{N_{X_i}} A_{i,j}, \qquad M_{\boldsymbol{A}} \equiv \bigwedge_{i=1}^{N} \bigwedge_{j=1}^{N_{X_i}} \bigwedge_{j' \neq j} \neg \left(A_{i,j} \wedge A_{i,j'}\right).$$

Then $C_{\boldsymbol{A}}$ is also a boolean function and $ran(d)$ is represented by $\{\boldsymbol{a} \mid \boldsymbol{a} \in \Omega_{\boldsymbol{A}}, C_{\boldsymbol{A}}(\boldsymbol{a}) = 1\}$. By the definition of $d$, $\boldsymbol{a} \notin ran(d)$ has no inverse image. To attach 0 probabilities to $\boldsymbol{a} \notin ran(d)$, we define $P_{\boldsymbol{A}}$ as follows:

$$P_{\boldsymbol{A}}(\boldsymbol{a};\boldsymbol{\theta}) \equiv C_{\boldsymbol{A}}(\boldsymbol{a}) \prod_{i=1}^{N} \prod_{j=1}^{N_{X_i}} \theta_{\pi_{i,j}}^{\boldsymbol{a}_{(i,j)}} = C_{\boldsymbol{A}}(\boldsymbol{a}) \prod_{i=1}^{M} \prod_{j=1}^{N_i} \theta_{i,j}^{\sigma_{i,j}(\boldsymbol{a})},$$

where $\sigma_{i,j}(\boldsymbol{a}) = |\{\boldsymbol{a}_{(i',j)} \mid \boldsymbol{a}_{(i',j)} = 1, \mu(i') = i\}|$. $P_{\boldsymbol{A}}(\boldsymbol{A};\boldsymbol{\theta})$ represents the same distribution as $P_{\boldsymbol{X}}(\boldsymbol{X};\boldsymbol{\theta})$ because $P_{\boldsymbol{A}}(\boldsymbol{A};\boldsymbol{\theta})$ satisfies

$$\forall \boldsymbol{x} \in \Omega_{\boldsymbol{X}} \left(P_{\boldsymbol{X}}(\boldsymbol{x};\boldsymbol{\theta}) = P_{\boldsymbol{A}}(d(\boldsymbol{x});\boldsymbol{\theta})\right), \qquad \forall \boldsymbol{a} \notin ran(d) \left(P_{\boldsymbol{A}}(\boldsymbol{a};\boldsymbol{\theta}) = 0\right).$$

Probabilistic events in $\mathcal{M}_{\boldsymbol{A}}$ are described by boolean formulas of $\boldsymbol{A}$. A probabilistic event $e$ is a subset of $\Omega_{\boldsymbol{A}}$. Let $F_e$ be a boolean formula representing $e$. $F_e$ satisfies $F_e(\boldsymbol{a}) = 1$ if $\boldsymbol{a} \in e$ otherwise $F_e(\boldsymbol{a}) = 0$. The probability of $e$ is computed as that of $F_e$ defined by

$$P_{\boldsymbol{A}}(F_e;\boldsymbol{\theta}) \equiv \sum_{\boldsymbol{a}\ s.t.\ F_e(\boldsymbol{a})=1} P_{\boldsymbol{A}}(\boldsymbol{a};\boldsymbol{\theta}) = \sum_{\boldsymbol{a} \in e} P_{\boldsymbol{A}}(\boldsymbol{a};\boldsymbol{\theta}),$$

where $P_{\boldsymbol{A}}(F_e;\boldsymbol{\theta})$ is a shorthand for $P_{\boldsymbol{A}}(F_e = 1;\boldsymbol{\theta})$.

We assume random variables in $\boldsymbol{X}$ are mutually independent. However, we can deal with variables which depend on other variables. Suppose $A$ and $B$ are boolean random variables and $B$ depends on $A$. A joint distribution of $A$ and $B$ is expressed as a product of the marginal distribution of $A$ and the conditional distribution of $B$ as follows:

$$P(A=a, B=b) = P(A=a)P(B=b \mid A=a), \qquad a, b \in \{0,1\}.$$

Using the above distribution $P$, $\boldsymbol{X}$ and $P_{\boldsymbol{X}}$ are defined as follows:

$$X \equiv \{A, B_0, B_1\}, \qquad P_{\boldsymbol{X}}(A=a) \equiv P(A=a), \qquad P_{\boldsymbol{X}}(B_a=b) \equiv P(B=b \mid A=a).$$

By the definition of $P_{\boldsymbol{X}}$, $P(A=a, B=b)$ is computed as $P_{\boldsymbol{X}}(A=a, B_a=b)$:

$$\begin{aligned} P_{\boldsymbol{X}}(A=a, B_a=b) &= P_{\boldsymbol{X}}(A=a)P_{\boldsymbol{X}}(B_a=b) \\ &= P(A=a)P(B=b \mid A=a) \\ &= P(A=a, B=b). \end{aligned}$$

Let's consider a simple example. A student $S$ goes to school by bus and he is sometimes late for school. There are only two causes of his being late. One is a traffic jam and the other is his oversleeping. We assume a traffic jam and his oversleeping are mutually independent but a traffic jam depends on the weather. Let $O$ be a boolean random variable corresponding to the occurrence of his oversleeping. Also let $W$ be a three-valued random variable representing the weather. $W$ takes one of $\{s, c, r\}$, where $s$, $c$ and $r$ are shorthands for *sunny*, *cloudy* and *rainy*, respectively. Let $J_w$ ($w \in \{s, c, r\}$) be a boolean random variable corresponding to a proposition that a traffic jam happens when the weather is $w$. By direct encoding, we get a set of boolean variables $\boldsymbol{A} \equiv \{\text{“}J_w = j\text{”}, \text{“}W = w\text{”}, \text{“}O = o\text{”} \mid w \in \{s, c, r\}, j, o \in \{0, 1\}\}$. Probabilistic events in this example are described by boolean formulas of $\boldsymbol{A}$. For example, a probabilistic event “$S$ is late for school” (“$S$ is not late for school”) is expressed as a boolean formula $F_L$ (resp. $F_{\bar{L}}$):

$$F_L \equiv \text{“}O = 1\text{”} \vee \left( \bigvee_{w \in \{s,c,r\}} \text{“}W = w\text{”} \wedge \text{“}J_w = 1\text{”} \right), \qquad F_{\bar{L}} \equiv \neg F_L.$$

A benefit of using logic is that various probabilistic events can be uniformly described by boolean formulas. For example, suppose we observe that $S$ is late for school on rainy day (resp. not rainy day). The observation is represented as $F_L \wedge \text{“}W = r\text{”}$ (resp. $F_L \wedge \neg\text{“}W = r\text{”}$).

Probabilistic events we observe are expressed as a sequence of boolean formulas. What we want to do in this paper is to estimate $\boldsymbol{\theta}$, a parameter of $\mathcal{M}_{\boldsymbol{A}}$, from the sequence. In the following subsection, we formulate the parameter estimation problem of $\mathcal{M}_{\boldsymbol{A}}$ and introduce an EM algorithm to solve it.

## 2.2 Parameter estimation

We here formulate our parameter estimation problem of $\mathcal{M}_{\boldsymbol{A}}$. Suppose we observe probabilistic events independently each other and describe them by boolean formulas. Let $\mathcal{O} \equiv \{F^{(t)}\}_{t=1}^{T}$ be a sequence representing our observations, where $F^{(t)}$ is a boolean formula which explains the $t$-th observation. The probability we observe $\mathcal{O}$ is

$$P_{\boldsymbol{A}}(\mathcal{O}; \boldsymbol{\theta}) = \prod_{t=1}^{T} P_{\boldsymbol{A}}(F^{(t)}; \boldsymbol{\theta}).$$

The goal of our parameter estimation problem is to find the *maximum likelihood estimate* $\boldsymbol{\theta}_{\mathrm{ML}}$ given by

$$\boldsymbol{\theta}_{\mathrm{ML}} \equiv \operatorname{argmax}_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{O}}(\boldsymbol{\theta}), \qquad \mathcal{L}_{\mathcal{O}}(\boldsymbol{\theta}) \equiv \ln P_{\boldsymbol{A}}(\mathcal{O}; \boldsymbol{\theta}),$$

where $\mathcal{L}_{\mathcal{O}}(\boldsymbol{\theta})$ is the *log likelihood function.* Note that observations $\mathcal{O}$ and assignments $\Omega_{\boldsymbol{A}}$ are not in one-to-one correspondence. This kind of observations is called *incomplete data* and the EM algorithm is known as a way to find the maximum likelihood estimator from incomplete data. The EM algorithm we develop for $\mathcal{M}_{\boldsymbol{A}}$ iterates the *expectation-step* (E-step) and the *maximization step* (M-step) until $\mathcal{L}_{\mathcal{O}}(\boldsymbol{\theta})$ converges. Each step is defined as follows.

- **E-step:** Compute *conditional expectations* $\eta_{\boldsymbol{\theta}}^{j}[i]$ $(1 \le i \le M, 1 \le j \le N_i)$ defined by

$$\eta_{\boldsymbol{\theta}}^{j}[i] \equiv \sum_{t=1}^{T} \sum_{\boldsymbol{a} \in \Omega_{\boldsymbol{A}}} \sigma_{i,j}(\boldsymbol{a}) P_{\boldsymbol{A}}(\boldsymbol{a} \mid F^{(t)}; \boldsymbol{\theta}). \tag{1}$$

- **M-step:** Update $\theta_{i,j}$ to $\theta_{i,j}^{new}$ by

$$\theta_{i,j}^{new} \equiv \eta_{\boldsymbol{\theta}}^{j}[i] / \textstyle\sum_{j'=1}^{N_i} \eta_{\boldsymbol{\theta}}^{j'}[i]. \tag{2}$$

The above EM algorithm still has a serious problem however. It is that the time complexity of computing conditional expectations $\eta_{\boldsymbol{\theta}}^{j}[i]$ defined by Eq. (1) is exponential in the size of $\boldsymbol{A}$. Fortunately, we need not to consider all of $\boldsymbol{a} \in \Omega_{\boldsymbol{A}}$ since $P_{\boldsymbol{A}}(\boldsymbol{a} \mid F^{(t)}; \boldsymbol{\theta})$ is 0 when $\boldsymbol{a}$ makes a boolean function $F^{(t)} \wedge C_{\boldsymbol{A}}$ false.

In the next section, we introduce BDDs and order encoding to represent $F^{(t)} \wedge C_{\boldsymbol{A}}$ compactly. A BDD is a rooted directed acyclic graph that gives a compact representation of a boolean function. Order encoding is another way to represent multi-valued variables by boolean variables. By combining them, we compress our observations as a graph and propose the BO-EM algorithm which is an EM algorithm for $\mathcal{M}_{\boldsymbol{A}}$ based on the graph.

## 3. Proposed algorithm

To execute the EM algorithm shown in Section 2.2 efficiently, we introduce BDDs and order encoding. Their combination compactly represents observations from $\mathcal{M}_{\boldsymbol{A}}$, and hence the probability computation and learning in $\mathcal{M}_{\boldsymbol{A}}$ are accelerated. In this section, we first briefly review BDDs and order encoding, then we propose the *BO-EM algorithm* which is an EM algorithm working on BDDs with order encoding.

### 3.1 Binary decision diagrams (BDDs)

Let $F$ be a boolean function of $\boldsymbol{A}$ and also let $\Delta_F$ be a *binary decision diagram* (BDD) for $F$, which is a rooted directed acyclic graph representing $F$. Every nodes in $\Delta_F$ is labeled by a variable in $\boldsymbol{A}$ or a constant $x \in \{0,1\}$. The label of a node $n$ is denoted by $Var(n)$. If $Var(n) = x$ $(x \in \{0,1\})$, $n$ is called the $x$-terminal, otherwise a *non-terminal*. Let $Nd(\Delta_F)$ be the set of all non-terminals. A non-terminal $n \in Nd(\Delta_F)$ has exactly two child nodes, the 1-child and the 0-child, denoted by $Ch_1(n)$ and $Ch_0(n)$, respectively. Let $\langle n, x\rangle$ $(x \in \{0,1\})$ be an edge from $n$ to $Ch_x(n)$. We call $\langle n, x\rangle$ the $x$-edge of $n$ and it indicates an assignment of $Var(n)$ is $x$. There exists an edge labeled by $F$, which has no parent node. The edge is denoted by $\langle F\rangle$ and called the *input edge*. A path from $\langle F\rangle$ to $x$-terminal represents a *partial assignment* of $\boldsymbol{A}$ making $F = x$, where the partial assignment is a set of assignments indicated by edges in the path. A set of all paths from $\langle F\rangle$ to 1-terminal represents all possible assignments of $\boldsymbol{A}$ making $F = 1$.

In general, there exist a lot of BDDs representing the same boolean function since some nodes in them can be deleted or merged by following two rules:

1. **Deletion rule:** Delete $n$ if $Ch_1(n) = Ch_0(n)$,
2. **Merging rule:** Merge $n$ and $n'$ if $Var(n) = Var(n')$ and $Ch_x(n) = Ch_x(n')$ $(x \in \{0,1\})$,

where $n$ and $n'$ are non-terminals. Applying these rules to nodes in a BDD never changes its target function. If $n \in Nd(\Delta_F)$ can be applied the above rule, $n$ is called a *redundant* node. When a BDD has no redundant node, it is said to be *reduced.* Also if each variable appears once in each path from the input edge to the terminal and if the order of the appearances is common to all these paths, it is said to be *ordered.* The common order is

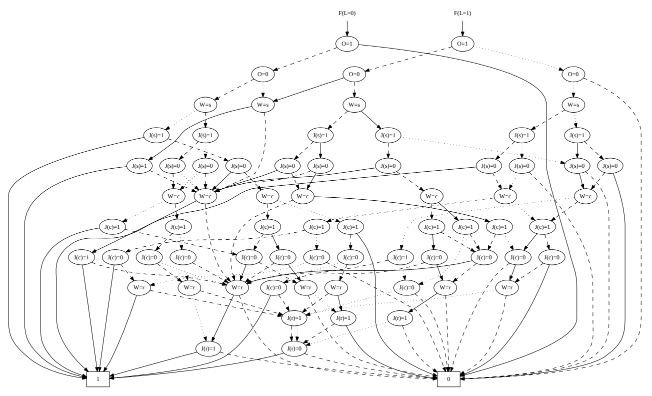

Figure 1: SBDD + direct encoding

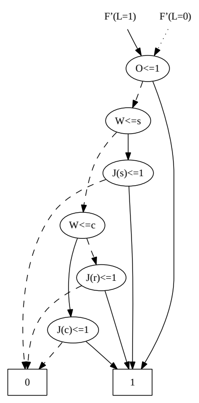

Figure 2: SBDD + order encoding

called *variable ordering* of the BDD. A reduced ordered BDD (ROBDD) is known to be a unique representation of the target boolean function (Akers (1978)).

An ordinary ROBDD represents a single boolean function. However multiple ROBDDs can be merged into a single diagram called a *shared* BDD (SBDD) (Minato et al. (1990)). Let $\mathcal{O} \equiv \{F_t\}_{t=1}^{T}$, where $F_t$ is a boolean function. Also let $\Delta_{\mathcal{O}}$ be an SBDD representing boolean functions $F_t \in \mathcal{O}$. $\Delta_{\mathcal{O}}$ has $T$ input edges $\langle F_t \rangle$ and a sub-graph pointed by $\langle F_t \rangle$ corresponds to a ROBDD of $F_t$. Sub-graphs which are complement each other in $\Delta_{\mathcal{O}}$ are merged by a *negative edge*. Pointing by a negative edge corresponds to the logical NOT operation. A path from $\langle F_t \rangle$ to the $x$-terminal represents a partial assignment of $\boldsymbol{A}$ making $F_t = x$ if the path includes even number of negative edges and otherwise $F_t = 1-x$. When a path represents $F_t = x$, we say that the path *logically* reaches the $x$-terminal. To provide uniqueness of SBDDs, the use of negative edges is limited to 1-edges or input edges. The size of an SBDD depends on target boolean functions and the variable ordering. To find the best variable ordering is NP-hard (Tani et al. (1996)). However, there are many efficient heuristics for good variable orderings (Minato et al. (1990)).

To execute the EM algorithm for $\mathcal{M}_{\boldsymbol{A}}$ efficiently, we represent our observations $\mathcal{O}$ as an SBDD $\Delta_F$ and compute conditional expectations $\eta^{j}_{\boldsymbol{\theta}}[i]$ on it. Sharing common structures in $\Delta_{\mathcal{O}}$ and computing probabilities/expectations on it correspond to sharing common probabilities/expectations between $F_1, \ldots, F_T$. As a result, we can avoid computing the same quantities over and over again. However, there still remains a serious problem that SBDDs and direct encoding are incompatible. Direct encoding represents a multi-valued random variable $X_i$ using boolean variables $A_{i,j}$ $(1 \leq j \leq N_{X_i})$. The values of $A_{i,j}$ and $A_{i,j}$ $(j' \neq j)$ are strongly related to each other. For example, when $A_{i,j}$ takes true, $A_{i,j'}$ must take false to make $C_{\boldsymbol{A}}$ true. Meanwhile, the deletion rule of BDDs deletes *don't care* nodes such that the value of the node is *don't care* in a path. Consequently, the deletion rule cannot delete nodes labeled by $A_{i,j'}$ following nodes labeled by $A_{i,j}$.

Figure 1 shows an SBDD for $F_L \wedge C_{\boldsymbol{A}}$ and $F_{\bar{L}} \wedge C_{\boldsymbol{A}}$. There are three types of edges, solid edges, dashed edges and dotted edges corresponding to 1-edges, 0-edges and negative edges, respectively. We can see that the SBDD is quite large. To simplify SBDDs, we introduce order encoding instead of direct encoding in the next subsection.

### 3.2 Order encoding

We here introduce *order encoding* and re-encode boolean formulas $F^{(t)} \in \mathcal{O}$ by it before we build up $\Delta_{\mathcal{O}}$ which is an SBDD representing $\mathcal{O}$. Order encoding represents $X_i$ which has an ordered discrete domain $\mathcal{D}_{X_i}$ by $N_{X_i}-1$ boolean variables. Let us define an order over $\mathcal{D}_{X_i}$ as $v_i^{(j)} < v_i^{(j')}$ if $j < j'$. Also let $\boldsymbol{B} \equiv \{B_{i,j} \mid 1 \le i \le N, 1 \le j \le N_{X_i}-1\}$, where $B_{i,j}$ is a boolean variable corresponding to a proposition "$X_i \le v_i^{(j)}$". By order encoding, $A_{i,j} \in \boldsymbol{A}$ is re-encoded to a boolean formula of $\boldsymbol{B}$:

$$A_{i,j} \text{ is encoded as } \begin{cases} B_{i,1} & j=1 \\ \left(\bigvee_{j'=1}^{j-1} \neg B_{i,j'}\right) \wedge B_{i,j} & 1<j<N_{X_i} \\ \bigvee_{j'=1}^{j-1} \neg B_{i,j'} & j = N_{X_i} \end{cases}.$$

The middle formula means that if $x_i \le v_i^{(j)}$ and $x_i > v_i^{(j')}$ for all $j' : 1 \le j' < j$ then $x_i$ must be $v_i^{(j)}$. In other words, when $B_{i,j}=1$ and $B_{i,j'}=0$ $(j' < j)$, we can say that $X_i = v_i^{(j)}$ without checking $B_{i,j'}$ $(j' > j)$. This property is strongly compatible with the deletion rule of BDDs. When the order of variables in $\Delta_{\mathcal{O}}$ is decided to be consistent with that over $\mathcal{D}_{X_i}$ $(1 \le i \le N)$, a path $\rho$ from an input edge to a terminal must satisfy the following two properties:

$$n_{i,j} \in \rho \implies \forall j' < j \left(\langle n_{i,j'}, 0\rangle \in \rho\right), \tag{3}$$

$$\langle n_{i,j}, 1\rangle \in \rho \implies \forall j'' > j \left(n_{i,j''} \notin \rho\right), \tag{4}$$

where $n_{i,j}$ is a node labeled by $B_{i,j}$. The property (3) says that a path through $n_{i,j}$ must contain 0-edges of $n_{i,j'}$ $(j' < j)$. The property (4) says that $n_{i,j''}$ under $n_{i,j}$ $(j'' > j)$ must be a redundant node and deleted by the deletion rule.

Figure 2 shows an SBDD representing $F_L$ and $F_{\bar{L}}$ re-encoded by order encoding. We can see that the SBDD becomes simple and $F_L$ and $F_{\bar{L}}$ are completely shared thanks to introducing order encoding. The combination of SBDDs and order encoding compresses observations from $\mathcal{M}_{\boldsymbol{A}}$ even if $\mathcal{M}_{\boldsymbol{A}}$ contains multi-valued random variables. In Section 4, we will empirically show that using order encoding reduces the complexity of the SBDD size for hidden Markov models.

### 3.3 BO-EM algorithm

We here propose the BO-EM algorithm which is an EM algorithm for $\mathcal{M}_{\boldsymbol{A}}$ working on an SBDD $\Delta_{\mathcal{O}}$ representing observations $\mathcal{O}$ from $\mathcal{M}_{\boldsymbol{A}}$ re-encoded by order encoding. First, we attach probabilities on edges in $\Delta_{\mathcal{O}}$, then we show how to compute $P_{\boldsymbol{A}}(F^{(t)}; \boldsymbol{\theta})$ and conditional expectations $\eta_{\boldsymbol{\theta}}^{j}[i]$ on $\Delta_{\mathcal{O}}$.

We now define probabilities of edges. Note that $B_{i,j}$ and $B_{i,j'}$ $(j \ne j')$ are dependent each other and we need to express this dependence in $\Delta_{\mathcal{O}}$. According to the property (3) in Section 3.2, a path including $n_{i,j}$ contains $\langle n_{i,j'}, 0\rangle$ $(j' < j)$. It means that $\langle n_{i,j}, x\rangle$ $(x \in \{0,1\})$ is conditioned by $\bigwedge_{j'=1}^{j-1} \neg B_{i,j'}$. Thus, we define the probability of $\langle n_{i,j}, x\rangle$ as

$$P_{\boldsymbol{A}}(\langle n_{i,j}, 1\rangle; \boldsymbol{\theta}) \equiv P_{\boldsymbol{A}}\left(B_{i,j} \mid \bigwedge_{j'=1}^{j-1} \neg B_{i,j'}; \boldsymbol{\theta}\right) = \frac{P_{\boldsymbol{A}}\left(B_{i,j} \wedge \left(\bigwedge_{j'=1}^{j-1} \neg B_{i,j'}\right); \boldsymbol{\theta}\right)}{P_{\boldsymbol{A}}\left(\bigwedge_{j'=1}^{j-1} \neg B_{i,j'}; \boldsymbol{\theta}\right)},$$

$$P_{\boldsymbol{A}}(\langle n_{i,j},0\rangle;\boldsymbol{\theta}) \equiv P_{\boldsymbol{A}}\left(\neg B_{i,j} \mid \textstyle\bigwedge_{j'=1}^{j-1} \neg B_{i,j'};\boldsymbol{\theta}\right) = \frac{P_{\boldsymbol{A}}\left(\bigwedge_{j'=1}^{j} \neg B_{i,j'};\boldsymbol{\theta}\right)}{P_{\boldsymbol{A}}\left(\bigwedge_{j'=1}^{j-1} \neg B_{i,j'};\boldsymbol{\theta}\right)}.$$

By the definition of order encoding, boolean formulas of $\boldsymbol{B}$ in the above probabilities are decoded as follows:

$$B_{i,j} \wedge \left(\textstyle\bigwedge_{j'=1}^{j-1} \neg B_{i,j'}\right) \Longrightarrow A_{i,j}, \qquad \textstyle\bigwedge_{j'=1}^{j-1} \neg B_{i,j'} \Longrightarrow \bigvee_{j'=j}^{N_{X_i}} A_{i,j'}.$$

By substituting the above into the definition of $P_{\boldsymbol{A}}(\langle n_{i,j},x\rangle;\boldsymbol{\theta})$ ($x \in \{0,1\}$), they are computed as follows:

$$P_{\boldsymbol{A}}(\langle n_{i,j},1\rangle;\boldsymbol{\theta}) = \frac{P_{\boldsymbol{A}}(A_{i,j};\boldsymbol{\theta})}{P_{\boldsymbol{A}}\left(\bigvee_{j'=1}^{N_{X_i}} A_{i,j'};\boldsymbol{\theta}\right)} = \frac{\theta_{\mu(i),j}}{\sigma_{\mu(i),j}},$$

$$P_{\boldsymbol{A}}(\langle n_{i,j},0\rangle;\boldsymbol{\theta}) = \frac{P_{\boldsymbol{A}}(\bigvee_{j'=j+1}^{N_{X_i}} A_{i,j'};\boldsymbol{\theta})}{P_{\boldsymbol{A}}(\bigvee_{j'=j}^{N_{X_i}} A_{i,j'};\boldsymbol{\theta})} = \frac{\sigma_{\mu(i),j+1}}{\sigma_{\mu(i),j}},$$

where $\sigma_{i,j} \equiv \sum_{j'=j}^{N_i} \theta_{i,j'}$ corresponds to the probability of $X_{i'} \geq v_{i'}^{(j)}$ ($\mu(i') = i$).

We next define probabilities of paths in $\Delta_{\mathcal{O}}$. Let $\rho$ be a path in $\Delta_{\mathcal{O}}$ from an input edge to a terminal and define its probability as follows:

$$P_{\boldsymbol{A}}(\rho;\boldsymbol{\theta}) \equiv \prod_{i=1}^{N} P_{\boldsymbol{A}}(\rho_i;\boldsymbol{\theta}), \qquad P_{\boldsymbol{A}}(\rho_i;\boldsymbol{\theta}) \equiv \prod_{\langle n,x\rangle \in \rho_i} P_{\boldsymbol{A}}(\langle n,x\rangle;\boldsymbol{\theta}),$$

where $\rho_i \equiv \{\langle n,x\rangle \mid \langle n,x\rangle \in \rho,\ \mathit{Var}(n) = B_{i,j}, 1 \leq j \leq N_i\}$. A partial path $\rho_i$ in $\rho$ indicates an assignment of $X_i$. When $\rho_i = \{\}$, the assignment of $X_i$ is *don't care* in $\rho$. In the case $\rho_i \neq \{\}$, $\rho_i$ indicates $X_i = v_i^{(j(\rho_i))}$ if $\rho_i$ has a 1-edge otherwise $X_i > v_i^{(j(\rho_i))}$, where $j(\rho_i) = \max\{j \mid \langle n,x\rangle \in \rho_i,\ \mathit{Var}(n) = B_{i,j}\}$. By the definition of probabilities of edges, it holds that:

$$P_{\boldsymbol{A}}(\rho_i;\boldsymbol{\theta}) = \begin{cases} 1 & \rho_i = \{\} \\ \theta_{\mu(i),j(\rho_i)} & \rho_i \text{ has a 1-edge} \\ \sigma_{\mu(i),j(\rho_i)+1} & \rho_i \text{ has no 1-edge} \end{cases}.$$

Consequently, $P_{\boldsymbol{A}}(\rho_i;\boldsymbol{\theta})$ corresponds to the probability $\rho_i$ indicates, and $P_{\boldsymbol{A}}(\rho;\boldsymbol{\theta})$ also corresponds to that of the assignment $\rho$ indicates.

A set of paths which logically reach the 1-terminal from an input edge $\langle F^{(t)}\rangle$ represents all possible assignments of $\boldsymbol{A}$ making $F^{(t)} = 1$. The sum of their probabilities corresponds to $P_{\boldsymbol{A}}(F^{(t)};\boldsymbol{\theta})$. To compute $P_{\boldsymbol{A}}(F^{(t)};\boldsymbol{\theta})$ on $\Delta_{\mathcal{O}}$ efficiently, we introduce *backward probabilities* $\mathcal{B}_{\boldsymbol{\theta}}^{x}[n]$ ($x \in \{0,1\}, n \in \mathit{Nd}(\Delta_{\mathcal{O}})$). $\mathcal{B}_{\boldsymbol{\theta}}^{x}[n]$ is a probability of paths logically reaching the $x$-terminal from $n$ and computed by dynamic programing on $\Delta_{\mathcal{O}}$ (Procedure 1). Let $\mathit{Root}(F^{(t)})$ be a node pointed by $\langle F^{(t)}\rangle$. $\mathcal{B}_{\boldsymbol{\theta}}^{x}[\mathit{Root}(F^{(t)})]$ is correspond to $P_{\boldsymbol{A}}(F^{(n)};\boldsymbol{\theta})$, where $x = 1$ if $\langle F^{(t)}\rangle$ is not a negative edge otherwise $x = 0$. All $P_{\boldsymbol{A}}(F^{(t)};\boldsymbol{\theta})$ ($F^{(t)} \in \mathcal{O}$) are computed by Procedure 1 all at once and its time/space complexity is $O(|\mathit{Nd}(\Delta_{\mathcal{O}})|)$.

**Procedure 1** Compute backward probabilities $\mathcal{B}^x_{\boldsymbol{\theta}}[n]$

```
1: $\mathcal{B}^1_{\boldsymbol{\theta}}[1] = 1,\ \mathcal{B}^0_{\boldsymbol{\theta}}[1] = 0,\ \mathcal{B}^1_{\boldsymbol{\theta}}[0] = 0,\ \mathcal{B}^0_{\boldsymbol{\theta}}[0] = 1$
2: **for** $i = |Nd(\Delta_{\mathcal{O}})|$ to 1, $x \in \{0,1\}$ **do**
3:     Let $n_i$ be the $i$-th node in the BDD ordered by topological order.
4:     **if** $\langle n_i, 1\rangle$ is not a negative edge **then**
5:         $\mathcal{B}^x_{\boldsymbol{\theta}}[n_i] = P_{\boldsymbol{A}}(\langle n_i, 0\rangle; \boldsymbol{\theta})\mathcal{B}^x_{\boldsymbol{\theta}}[Ch_0(n_i)] + P_{\boldsymbol{A}}(\langle n_i, 1\rangle; \boldsymbol{\theta})\mathcal{B}^x_{\boldsymbol{\theta}}[Ch_1(n_i)]$
6:     **else if** $\langle n_i, 1\rangle$ is a negative edge **then**
7:         $\mathcal{B}^x_{\boldsymbol{\theta}}[n_i] = P_{\boldsymbol{A}}(\langle n_i, 0\rangle; \boldsymbol{\theta})\mathcal{B}^x_{\boldsymbol{\theta}}[Ch_0(n_i)] + P_{\boldsymbol{A}}(\langle n_i, 1\rangle; \boldsymbol{\theta})\mathcal{B}^{1-x}_{\boldsymbol{\theta}}[Ch_0(n_i)]$
8:     **end if**
9: **end for**
```

At the last, we show how to compute conditional expectations $\eta^j_{\boldsymbol{\theta}}[i]$ on $\Delta_{\mathcal{O}}$. By the definition of $\eta^j_{\boldsymbol{\theta}}[i]$, it can be computed as $\sum_{X_{i'} \in \pi_i} \eta^i_{\boldsymbol{\theta}}[X_{i'}]$, where $\eta^j_{\boldsymbol{\theta}}[X_i]$ is defined by

$$\eta^j_{\boldsymbol{\theta}}[X_i] \equiv \sum_{t=1}^{T} \sum_{\boldsymbol{a} \in \Omega_{\boldsymbol{A}}} \boldsymbol{a}_{(i,j)} P_{\boldsymbol{A}}(\boldsymbol{a} \mid F^{(t)}; \boldsymbol{\theta}) = \sum_{t=1}^{T} P_{\boldsymbol{A}}(A_{i,j} \mid F^{(t)}; \boldsymbol{\theta}),$$

where $P_{\boldsymbol{A}}(A_{i,j} \mid F^{(t)}; \boldsymbol{\theta})$ corresponds to a conditional probability of $X_i = v^{(j)}_i$ given $F^{(t)}$. Paths in $\Delta_{\mathcal{O}}$ are classified into three types by an assignment of $X_i$ in them: (a) $X_i = v^{(j)}_i$, (b) $X_i > v^{(j)}_i$, (c) $X_i$ is *don't care*. To compute $\eta^j_{\boldsymbol{\theta}}[i]$, we extract $P_{\boldsymbol{A}}(A_{i,j} \mid F^{(t)}; \boldsymbol{\theta})$ from the above three types of paths. We introduce *forward expectations* $\mathcal{F}^x_{\boldsymbol{\theta}}[n]$ $(x \in \{0,1\}, n \in Nd(\Delta_{\mathcal{O}}))$ computed by dynamic programming on $\Delta_{\mathcal{O}}$ like backward probabilities (Procedure 2). $\mathcal{F}^1_{\boldsymbol{\theta}}[n]$ (resp. $\mathcal{F}^0_{\boldsymbol{\theta}}[n]$) corresponds to a weighted sum of probabilities of paths from $\langle F^{(t)}\rangle$ to $n$ including even (resp. odd) number of negative edges, where the weight is the inverse of $P_{\boldsymbol{A}}(F^{(t)}; \boldsymbol{\theta})$. Using $\mathcal{F}^x_{\boldsymbol{\theta}}[n]$ and $\mathcal{B}^x_{\boldsymbol{\theta}}[n]$, we compute various conditional probabilities. For instance, $\sum_{x \in \{0,1\}} \mathcal{F}^x_{\boldsymbol{\theta}}[n]\mathcal{B}^x_{\boldsymbol{\theta}}[n]$ corresponds to a conditional probability of paths including $n$ given $F^{(t)}$. In a similar manner, conditional expectations $\eta^j_{\boldsymbol{\theta}}[i]$ are computed by Procedure 3 and its time/space complexity is $O(|Nd(\Delta_{\mathcal{O}})|)$.

Summarizing the above, the E-step of BO-EM consists of the following three steps whose time/space complexity is $O(|Nd(\Delta_{\mathcal{O}})|)$.

1. Compute *backward probabilities* $\mathcal{B}^x_{\boldsymbol{\theta}}[n]$ $(n \in Nd(\Delta_{\mathcal{O}}), x \in \{0,1\})$.
2. Compute *forward expectations* $\mathcal{F}^x_{\boldsymbol{\theta}}[n]$ $(n \in Nd(\Delta_{\mathcal{O}}), x \in \{0,1\})$.
3. Compute *conditional expectations* $\eta^j_{\boldsymbol{\theta}}[i]$ $(1 \le i \le N, 1 \le j \le N_i)$.

In the M-step, BO-EM updates $\boldsymbol{\theta}$ following Fq. (2). It repeats these two steps until $\mathcal{L}_{\mathcal{O}}(\boldsymbol{\theta})$ converges and outputs the final $\boldsymbol{\theta}$ as estimated parameters.

## 4. Experiments

We first apply BO-EM to learning parameters of *hidden Markov models* (HMMs) (Rabiner (1989)) to show the combination of SBDDs and order encoding compresses observations from HMMs compactly. We next utilize BO-EM to give a diagnosis for failure in logic circuits which involve some stochastic error gates.

**Procedure 2** Compute forward expectations $\mathcal{F}^x_{\boldsymbol{\theta}}[n]$

```
1: $\mathcal{F}^x_{\boldsymbol{\theta}}[n] = 0$ for all $n \in Nd(\Delta_{\mathcal{O}}), x \in \{0,1\}$
2: for $t = 1$ to $T$ do
3:     $n = Root(F^{(t)})$
4:     if $\langle F^{(t)} \rangle$ is not a negative edge then
5:         $\mathcal{F}^1_{\boldsymbol{\theta}}[n] {+}{=} 1/\mathcal{B}^0_{\boldsymbol{\theta}}[n]$
6:     else if $\langle F^{(t)} \rangle$ is a negative edge then
7:         $\mathcal{F}^0_{\boldsymbol{\theta}}[n] {+}{=} 1/\mathcal{B}^0_{\boldsymbol{\theta}}[n]$
8:     end if
9: end for
10: for $i = 1$ to $|Nd(\Delta_{\mathcal{O}})|$, $x \in \{0,1\}$ do
11:     Let $n_i$ be the $i$-th node in the BDD ordered by topological order.
12:     $\mathcal{F}^x_{\boldsymbol{\theta}}[Ch_0(n)] {+}{=} \mathcal{F}^x_{\boldsymbol{\theta}}[n_i] P_{\boldsymbol{A}}(\langle n_i, 0 \rangle; \boldsymbol{\theta})$
13:     if $\langle n_i, 1 \rangle$ is not a negative edge then
14:         $\mathcal{F}^x_{\boldsymbol{\theta}}[Ch_1(n)] {+}{=} \mathcal{F}^x_{\boldsymbol{\theta}}[n_i] P_{\boldsymbol{A}}(\langle n_i, 1 \rangle; \boldsymbol{\theta})$
15:     else if $\langle n_i, 1 \rangle$ is a negative edge then
16:         $\mathcal{F}^x_{\boldsymbol{\theta}}[Ch_1(n)] {+}{=} \mathcal{F}^{1-x}_{\boldsymbol{\theta}}[n_i] P_{\boldsymbol{A}}(\langle n_i, 1 \rangle; \boldsymbol{\theta})$
17:     end if
18: end for
```

### 4.1 BO-EM for HMMs

We here apply the BO-EM algorithm to learn probabilities of HMMs which define a probability distribution over strings. An HMM has *hidden state* $S_t$ following a Markov chain and stochastically outputs a symbol $O_t$ depending on $S_t$ at time slice $t$. As the result of repeating it $L$ times, the HMM generates a string consisting of $L$ symbols. Let $s_t$ and $o_t$ be values of $S_t$ and $O_t$, respectively. For simplicity, we assume $s_t \in \{1, \ldots, N\}$ and $o_t \in \{1, \ldots, M\}$, where $N$ and $M$ is the number of states and symbols, respectively. The HMM has three types of parameters $\pi_i \equiv P(S_1 = i)$, $a_{i,j} \equiv P(S_t = j \mid S_{t-1} = i)$ and $b_{i,k} \equiv P(O_t = k \mid S_t = i)$ $(1 \le i, j \le N, 1 \le k \le M, 1 \le t \le L)$. Let $O \equiv \{o_t\}_{t=1}^L$ be a string and $S \equiv \{s_t\}_{t=1}^L$ be a state sequence. The HMM defines a joint distribution of $O$ and $S$ such that

$$P(s_1, o_1, \ldots, s_L, o_L) \equiv \pi_{s_1} b_{s_1,o_1} \prod_{t=2}^{L} a_{s_{t-1},s_t} b_{s_t,o_t}.$$

The above distribution $P$ can be represented by $P_{\boldsymbol{X}}$ which is a joint distribution over $\boldsymbol{X}$, where $\boldsymbol{X}$ is a set of independent random variables. We define $\boldsymbol{X} \equiv \{S_1, S_t^{(i)}, O_t^{(i)} \mid 1 \le i \le N, 1 \le t \le L\}$, $P_{\boldsymbol{X}}(S_1 = i) \equiv \pi_i$, $P_{\boldsymbol{X}}(S_t^{(j)} = j) \equiv a_{i,j}$ and $P_{\boldsymbol{X}}(O_t^{(i)} = k) \equiv b_{i,k}$. The above probability $P(s_1, o_1, \ldots, s_L, o_L)$ is computed as $P_{\boldsymbol{X}}(s_1, o_1^{(s_1)}, \ldots, s_L^{(s_{L-1})}, o_t^{(s_L)})$:

$$P_{\boldsymbol{X}}(s_1) P_{\boldsymbol{X}}(o_1^{(s_1)}) \prod_{t=2}^{L} P_{\boldsymbol{X}}(s_t^{(s_{t-1})}) P_{\boldsymbol{X}}(o_t^{(s_t)}) = \pi_{s_1} b_{s_1,o_1} \prod_{t=2}^{L} a_{s_{t-1},s_t} b_{s_t,o_t},$$

where $s_t^{(s_{t-1})}$ and $o_t^{(s_t)}$ are shorthands for $S_t^{(s_{t-1})} = s_t$ and $O_t^{(s_t)} = o_t$, respectively. Let $\mathcal{M}_{\boldsymbol{A}}$ be a propositional logic-based probabilistic model defined by propositionalizing $\boldsymbol{X}$ and $P_{\boldsymbol{X}}$.

**Procedure 3** Compute conditional expectations $\eta^j_{\boldsymbol\theta}[i]$

```
1: $\eta^j_{\boldsymbol\theta}[i]=0$, $\eta^j_{\boldsymbol\theta}[X_i]=0$, $\gamma^j[X_i]=0$, $\mathcal{D}[B_{i,j}]=0$ for all $1\le i\le N, 1\le j\le N_{X_i}$
2: for all $n\in Nd(\Delta_{\mathcal{O}})$
3:    $B_{i,j}=Var(n)$, $B_1=Var(Ch_1(n))$, $B_0=Var(Ch_0(n))$, $B_N=next_{\mathcal{O}}(B_{i,j})$
4:    $e_0=\mathcal{F}^0_{\boldsymbol\theta}[n]P_{\boldsymbol A}(\langle n,0\rangle;\boldsymbol\theta)\mathcal{B}^0_{\boldsymbol\theta}[Ch_0(n)]+\mathcal{F}^1_{\boldsymbol\theta}[n]P_{\boldsymbol A}(\langle n,0\rangle;\boldsymbol\theta)\mathcal{B}^1_{\boldsymbol\theta}[Ch_0(n)]$
5:    if $\langle n,1\rangle$ is not a negative edge then
6:       $e_1=\mathcal{F}^0_{\boldsymbol\theta}[n]P_{\boldsymbol A}(\langle n,1\rangle;\boldsymbol\theta)\mathcal{B}^0_{\boldsymbol\theta}[Ch_1(n)]+\mathcal{F}^1_{\boldsymbol\theta}[n]P_{\boldsymbol A}(\langle n,1\rangle;\boldsymbol\theta)\mathcal{B}^1_{\boldsymbol\theta}[Ch_1(n)]$
7:    else if $\langle n,1\rangle$ is a negative edge then
8:       $e_1=\mathcal{F}^0_{\boldsymbol\theta}[n]P_{\boldsymbol A}(\langle n,1\rangle;\boldsymbol\theta)\mathcal{B}^1_{\boldsymbol\theta}[Ch_1(n)]+\mathcal{F}^1_{\boldsymbol\theta}[n]P_{\boldsymbol A}(\langle n,1\rangle;\boldsymbol\theta)\mathcal{B}^0_{\boldsymbol\theta}[Ch_1(n)]$
9:    end if
10:   $\eta^j_{\boldsymbol\theta}[X_i]$+= $e_1$                                  /* Add the expectation of (a). */
11:   $\gamma^{j+1}[X_i]$+= $e_0$, $\gamma^j[X_i]$−= $e_0+e_1$                 /* Store the expectation of (b). */
12:   $\mathcal{D}[B_N]$+= $e_0+e_1$, $\mathcal{D}[B_1]$−= $e_1$, $\mathcal{D}[B_0]$−= $e_0$   /* Store the expectation of (c). */
13: end for
14: $tmp=0$
15: for $k=2$ to $|\boldsymbol B|$ do
16:    Let $B^{(k)}$ be the $k$-th variable in the variable order of $\Delta_{\mathcal{O}}$.
17:    $tmp$+= $\mathcal{D}[B^{(k)}]$
18:    if $B^{(k)}=B_{i,1}$ then
19:       $\gamma^1[X_i]=tmp$
20:    end if
21: end for
22: for $i=1$ to $N$ do
23:    $tmp=0$
24:    for $j=1$ to $N_{X_i}$ do
25:       $tmp$+= $\gamma^j[X_i]/\sigma_{\mu(i),j}$
26:       $\eta^j_{\boldsymbol\theta}[X_i]$+= $tmp*\theta_{\mu(i),j}$                /* Add stored expectation of (b) & (c). */
27:    end for
28: end for
29: $\eta^j_{\boldsymbol\theta}[i]$+= $\eta^j_{\boldsymbol\theta}[X_i]$ for all $X_{i'}\in\pi_i, 1\le j\le N_{X_i}$     /* Sum up $\eta^j_{\boldsymbol\theta}[X_{i'}]$ in $\eta^j_{\boldsymbol\theta}[i]$. */
```

$\mathcal{M}_{\boldsymbol A}$ represents the HMM and a string $O$ is explained as a boolean formula $F_O$:

$$F_O \equiv \bigvee_{i=1}^N \left(\text{“}S_1=i\text{”} \wedge \text{“}O_1^{(i)}=o_1\text{”} \wedge F_O^{(2,i)}\right),$$
$$F_O^{(t,i)} \equiv \begin{cases} \bigvee_{j=1}^N \left(\text{“}S_t^{(i)}=j\text{”} \wedge \text{“}O_t^{(j)}=o_t\text{”} \wedge F_O^{(t+1,j)}\right) & 2\le t\le L \\ true & t>L \end{cases}.$$

We now fix $M=2$ and $L=5$. Then, the number of possible strings generated by the HMM is 32. We build up two SBDDs representing these 32 strings, one uses order encoding and the other uses direct encoding. Figure 3 depicts the sizes of them as a function of $N$. Variable ordering is decided by expanding the best one in the case $L=2$ and $N=2$. The graph shows the ratio between (a) and (b) is proportional to $N$. More precisely, the size of the SBDD representing $F_O$ decreases from $O(N^3L)$ to $O(N^2L)$ by using order encoding. Consequently, the time/space complexity of BO-EM for HMMs also become $O(N^2L)$ and it is the same as that of the EM algorithm specialized for HMMs.

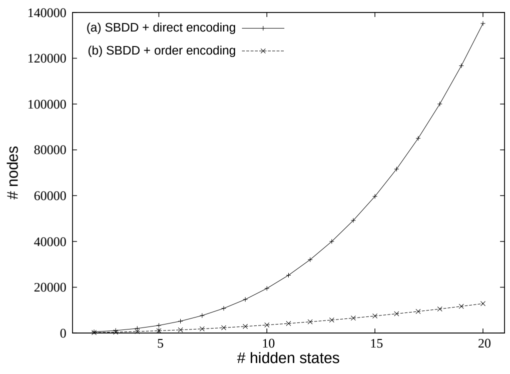

Figure 3: SBDD size

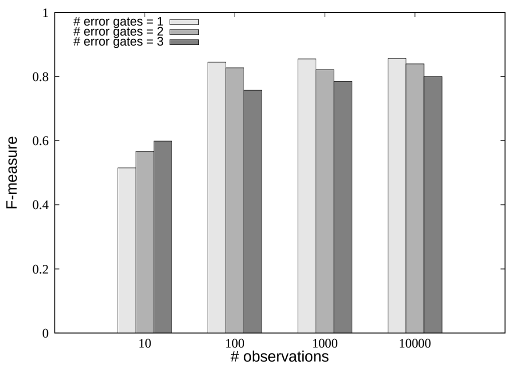

Figure 4: F-measure

## 4.2 BO-EM to give a diagnosis for failure in logic circuits

We here utilize BO-EM to give a diagnosis for failure in a logic circuit involving error gates. The task is to find error gates in the logic circuit from observations which are pairs of its input and output values. We assume an error gate is stochastically stuck-at 0 or 1. To handle stochastic errors, we introduce a random variable $st(G)$ representing a state of the gate $G$ which takes one of $\{ok, stk0, stk1\}$. $st(G) = ok$ means $G$ works well. And $st(G)=stk0$ (resp. $stk1$) means its output is stuck-at 0 (resp. 1). We learn probabilities of $st(G)$ from observations using BO-EM and give a diagnosis that $G$ is an error gate if the probability of $st(G)=ok$ is low.

To get boolean formulas explaining observations, we use PRISM (Sato and Kameya (2001)) which is a logic-based probabilistic modeling language for generative modeling. A logic circuit can be defined by two predicates, $type(G,T)$ and $conn(P,Q)$. $type(G,T)$ means that $G$'s gate type is $T$. $conn(P,Q)$ represents $P$ and $Q$ are connected. As an illustrative example, let's consider a circuit $C$ representing $(I_1 \wedge I_2) \vee I_3$. Let $i(N)$ be the $N$-th input $I_N$. Also let $o(c)$ be the output of $C$. Then, $C$ is defined as follows:

$$\begin{array}{lll} type(g1, and). & conn(i(1), in(1, g1)). & conn(out(g1), in(2, g2)). \\ type(g2, or). & conn(i(2), in(2, g1)). & conn(out(g2), o(c)). \\ & conn(i(3), in(1, g2)). & \end{array}$$

where $in(I,G)$ is the $I$-th input of $G$ and $out(G)$ is the output of $G$. We next define values of gates. To handle probabilistic events, PRISM has a particular predicate $msw$ with arity 2. For example, $msw(st(G),S)$ represents that a random variable $st(G)$ takes a value $S$. We define a predicate $val(P,V)$ which represents the value of $P$ being $V$ as follows:

$$\begin{aligned} val(Q,V) &: -conn(P,Q), val(Q,V). \\ val(out(G),V) &: -msw(st(G),S), (S=ok, normal(G,V);\ S=stk0, V=0;\ S=stk1, V=1). \\ normal(G,V) &: -type(G,T), (T=or, or(G,V);\ T=and, and(G,V)). \\ or(G,V) &: -(val(in(1,G),1), V=1;\ val(in(2,G),1), V=1 \\ &\quad ;val(in(1,G),0), val(in(2,G),0), V=0). \\ and(G,V) &: -(val(in(1,G),0), V=0;\ val(in(2,G),0), V=0 \\ &\quad ;val(in(1,G),1), val(in(2,G),1), V=1). \end{aligned}$$

Using the above PRISM program, we can derive boolean formulas which explain probabilistic events in the circuit. For example, a probabilistic event $val(out(g2),1)$ is expressed as the following formula:

$$\begin{aligned} val(out(g2),1) \Leftrightarrow\ & msw(st(g2),stk1) \vee (val(in(1,g2),1) \wedge msw(st(g2),ok)) \\ & \vee\ (val(in(2,g2),1) \wedge msw(st(g2),ok)) \,. \end{aligned}$$

PRISM requires the exclusive condition that the disjuncts in formulas are probabilistically exclusive to make sum-product probability computation possible. However, the first and the second disjunct in the above formula are not probabilistically exclusive.

As an example of eliminating the exclusiveness condition of PRISM, we give a diagnosis of a 3-bit adder circuit $C_{3ad}$ using BO-EM and boolean formulas derived by the above PRISM program. The circuit consists of 12 gates, and $E$ gates out of them are error gates. The number of possible circuit states and possible diagnoses are $3^{12}$ and $2^{12}$, respectively. We give a diagnosis using probabilities of $st(G)$ learned by BO-EM . The detail of experiment settings is as follows:

1. Generate a 3-bit adder circuit $C_{3ad}$ involving $E$ error gates.
2. Sample $N$ observations from $C_{3ad}$ and derive boolean formulas explaining them by the above PRISM program.
3. Learn probabilities of $st(G)$ by BO-EM from formulas generated in Step 2.
4. Give a diagnosis using the learned probabilities: If $P(st(G) = ok) \leq 0.5$ then $G$ is judged as an error gate otherwise a normal gate.
5. Repeat (1)-(4) 100 times and compute the F-measure.

In Step 1, we fix $P(st(G)=ok)$ at 1 if $G$ is a normal gate, otherwise 0. $P(st(G)=stk0)$ and $P(st(G)=stk1)$ are sampled from a uniform distribution. The F-measure computed in Step 5 is a harmonic average of *recall* and *precision*. Recall is the rate of gates judged as errors in true error gates. Precision is the rate of true error gates in gates judged as errors. Figure 4 depicts the F-measure as a function of $E$ and $N$. The graph shows diagnostic accuracy of each $E$ increases with $N$ and BO-EM gives diagnoses with highest diagnostic accuracy when $N = 10000$. The result suggests BO-EM is useful to give a diagnosis for failure in logic circuits involving stochastic error gates.

## 5. Related work

A propositional logic-based probabilistic model $\mathcal{M}_{\boldsymbol{A}}$ can be represented as a discrete Bayesian network (BN) with parameter tying, and various encoding techniques and BDD-like data structures for efficient probabilistic inference in BNs have been proposed (Chavira and Darwiche (2008), Mateescu and Dechter (2007)). The difference between these existing approaches and BO-EM is that the former specializes in probabilistic inference whereas the latter focuses on parameter learning where it is required to compute conditional expectations for parameters. A conditional expectation is calculated as a sum of conditional probabilities for each observation. However, computations of conditional probabilities for multiple parameters and observations oftentimes overlap and hence cause redundancy. To avoid such redundancy, BO-EM computes every conditional expectation all in once on an

SBDD sharing their fragments of common probabilities and expectations. Consequently, BO-EM can make parameter learning for BNs efficiently. Similarly, it is expected that BO-EM also makes efficient parameter learning in relational Bayesian networks (RBNs) (Jaeger (1997)) and Bayesian logic programs (BLPs) (Getoor and Taskar (2007)).

There already exist BDD-based probability learning algorithms for LBPMs. Ishihata et al. (2008) proposed the BDD-EM algorithm and Gutmann et al. (2010) proposed the CoPrEM algorithm for ProbLog (De Raedt et al. (2007)) which is a recent logic-based formalism that computes probabilities via BDDs. They assume a target probability distribution to be learned is a product of Bernoulli distribution. Because of the assumption, they cannot deal efficiently with multi-valued random variables. BO-EM has two important extensions compared to them. The first one is that it copes with multi-valued random variables by introducing order encoding. The second one is that it adopts SBDDs involving negative edges not in ordinary BDDs. Using an SBDD and negative edges compresses observations into a single diagram and makes their common sub-graphs shared. Meanwhile, BDD-EM adopts *decomposed* BDDs (DBDDs) which are a kind of hierarchical BDDs to share common partial functions between BDDs. To realize probability learning on DBDDs, BDD-EM computes *inside probabilities* and *outside expectations* which are similar to those of probabilistic context free grammars (PCFGs). We can introduce DBDDs to BO-EM and it enables us to learn probabilities of PCFGs in the same complexity as the EM algorithm specialized for them.

## 6. Conclusion

We have proposed the BO-EM algorithm which is an EM algorithm for propositional logic-based probabilistic models $\mathcal{M}_{\boldsymbol{A}}$. It adopts SBDDs and order encoding to compress observations from $\mathcal{M}_{\boldsymbol{A}}$ described by boolean formulas. As a result, they accelerate probability computation and learning in $\mathcal{M}_{\boldsymbol{A}}$. BO-EM is generic in the sense that it is applicable to SBDDs representing any boolean formulas, and at the same time can be efficient thanks to dynamic programing on the SBDD. Its actual computation time depends on the SBDD size. For instance, when we learn parameters of an HMM by BO-EM , the SBDD size and its complexity is $O(N^2L)$. It is the same as that of *Baum-Welch* algorithm which is the EM algorithm specialized for HMMs.

As shown in Section 4.2, BO-EM can solve a long-standing problem of PRISM. It employs data structure called *explanation graph*s representing boolean formulas in disjunctive normal form and assumes the exclusiveness condition that the disjuncts are exclusive to make sum-product probability computation possible. Since BO-EM is applicable to explanation graphs as well, it allows PRISM to abolish the exclusiveness condition entirely.

BO-EM is useful for real application domains also. Inoue et al. (2009) applied BDD-EM to evaluating hypotheses about metabolic pathways. They assume propositional variables in their logical model to be independent boolean random variables. On the other hands, Synnaeve et al. (2009) proposed another logical model about metabolic pathway containing multi-valued variables. In their model, concentrations of metabolites are discretized into three levels. Replacing BDD-EM with BO-EM in the system Inoue et al. (2009) proposed enables us to evaluate hypotheses generated from models which involve multi-valued variables like the one Synnaeve et al. (2009) proposed.

## References

Sheldon B. Akers. Binary decision diagrams. *IEEE Transaction on Computers*, 27(6): 509–516, 1978.

Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172:81–129, 2008.

Luc De Raedt, Angelika Kimming, and Hannu Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *Proc. of IJCAI'07*, pages 2468–2473, 2007.

Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors. *Probabilistic Inductive Logic Programming - Theory and Applications*, 2008. Springer.

Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning).* The MIT Press, 2007. ISBN 0262072882.

Bernd Gutmann, Ingo Thon, and Luc De Raedt. Learning the parameters of probabilistic logic programs from interpretations. Department of Computer Science, K.U.Leuven, 2010.

Katsumi Inoue, Taisuke Sato, Masakazu Ishihata, Yoshitaka Kameya, and Hidetomo Nabeshima. Evaluating abductive hypotheses using an EM algorithm on BDDs. In *Proc. of IJCAI'09*, pages 810–815, 2009.

Masakazu Ishihata, Yoshitaka Kameya, Taisuke Sato, and Shin-ichi Minato. Propositionalizing the EM algorithm by BDDs. *TR08-0004, Dept. of Computer Science, Tokyo Institute of Technology*, 2008.

Manfred Jaeger. Relational bayesian networks. In *Proc. of UAI'97*, pages 266–273, 1997.

Robert Mateescu and Rina Dechter. And/or multi-valued decision diagrams (AOMDDs) forweighted graphical models. In *Proc. of UAI'07*, 2007.

Shin-ichi Minato, Nagasa Ishiura, and Shuzo Yajima. Shared binary decision diagram with attributed edges for efficient boolean function manipulation. In *Proc. of DAC'90*, pages 52–57, 1990.

Stephen Muggleton. Stochastic logic programs. In *New Generation Computing.* Academic Press, 1996.

David Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proc. of IEEE*, pages 257–286, 1989.

Taisuke Sato and Yoshitaka Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

Gabriel Synnaeve, Andrei Doncescu, and Katsumi Inoue. Kinetic models for logic-based hypothesis finding in metabolic pathways. In *Proc. of ILP'09*, 2009.

Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, and Mutsunori Banbara. Compiling finite linear CSP into SAT. In *Proc. of CP'06*, pages 590–603, 2006.

Seiichiro Tani, Kiyoharu Hamaguchi, and Shuzo Yajima. The complexity of the optimal variable ordering problems of a shared binary decision diagram. *IEICE Transactions on Information and Systems*, 79(4):271–281, 1996.