

Errata to Foundations of Probabilistic Logic Programming

Fabrizio Riguzzi

Page 1

Replace

An element $a \in S$ is the *least upper bound* of a subset X of X

with

An element $a \in S$ is the *least upper bound* of a subset X of S

Page 2

Replace

A relation $<$ defined by $a < b$ iff $a < b$ and $a \neq b$ is associated with any partial order \leq on S .

with

A relation $<$ defined by $a < b$ iff $a \leq b$ and $a \neq b$ is associated with any partial order \leq on S .

Page 17

Replace **Definition 2** ($OpFalse_{\mathcal{I}}^P$ and $OpFalse_{\mathcal{I}}^P$ operators) with **Definition 2** ($OpTrue_{\mathcal{I}}^P$ and $OpFalse_{\mathcal{I}}^P$ operators)

Page 25

Replace $P(X = \omega)$ with $P(X \in \omega)$.

Page 73

Replace w in the first line with w_{σ}

Page 107

The last formula of the proof of Lemma 7 should be replaced by

$$WFM(w_\sigma) = WFM(w_\sigma | \mathcal{I}_\alpha) = WFM(w_\sigma | IFPP^{\mathcal{P}} \uparrow \alpha).$$

The proof of Lemma 8 should be:

This is a simple consequence of Lemma 7: $w_\sigma \in \omega_{K_\alpha^a}$ means that a is a fact in $w_\sigma | IFPP^{\mathcal{P}} \uparrow \alpha$ so $WFM(w_\sigma | IFPP^{\mathcal{P}} \uparrow \alpha) \models a$ and $WFM(w_\sigma) \models a$.

On the other hand, $w_\sigma \in \omega_{K_\alpha^a}$ means that there are not rules for a in $w_\sigma | IFPP^{\mathcal{P}} \uparrow \alpha$ therefore $WFM(w_\sigma | IFPP^{\mathcal{P}} \uparrow \alpha) \models \sim a$ and $WFM(w_\sigma) \models \sim a$.

Page 117

Formula

$$\begin{aligned} P(q) &= \int_{\sigma \in S_{\mathcal{P}}, \mathbf{x} \in \mathbb{R}^n} p(q, w_{\sigma, \mathbf{x}}) = \\ &= \int_{\sigma \in S_{\mathcal{P}}, \mathbf{x} \in \mathbb{R}^n} P(q | w_{\sigma, \mathbf{x}}) p(w_{\sigma, \mathbf{x}}) = \\ &= \int_{\sigma \in S_{\mathcal{P}}, \mathbf{x} \in \mathbb{R}^n : w_{\sigma, \mathbf{x}} \models q} p(w_{\sigma, \mathbf{x}}) \end{aligned}$$

should be replaced by

$$\begin{aligned} P(q) &= \int_{\sigma \in S_{\mathcal{P}}, \mathbf{x} \in \mathbb{R}^n} p(q, w_{\sigma, \mathbf{x}}) d\sigma d\mathbf{x} = \\ &= \int_{\sigma \in S_{\mathcal{P}}, \mathbf{x} \in \mathbb{R}^n} P(q | w_{\sigma, \mathbf{x}}) p(w_{\sigma, \mathbf{x}}) d\sigma d\mathbf{x} = \\ &= \int_{\sigma \in S_{\mathcal{P}}, \mathbf{x} \in \mathbb{R}^n : w_{\sigma, \mathbf{x}} \models q} p(w_{\sigma, \mathbf{x}}) d\sigma d\mathbf{x} = \\ &= \sum_{\sigma \in S_{\mathcal{P}}} \int_{\mathbf{x} \in \mathbb{R}^n : w_{\sigma, \mathbf{x}} \models q} p(w_{\sigma, \mathbf{x}}) d\mathbf{x} \end{aligned}$$

Page 154

The text:

Binary Decision Diagrams (BDDs) perform a Shannon expansion of the Boolean formula: they express the formula as

$$f_K(\mathbf{X}) = X_1 \vee f_K^{X_1}(\mathbf{X}) \wedge \neg X_1 \vee f_K^{\neg X_1}(\mathbf{X})$$

should be replaced by

BDDs perform a Shannon expansion of the Boolean formula: they express the formula as

$$f_K(\mathbf{X}) = X_1 \wedge f_K^{X_1}(\mathbf{X}) \vee \neg X_1 \wedge f_K^{\neg X_1}(\mathbf{X})$$

Page 161

The text:

The Boolean variables are associated with the following parameters:

$$\begin{aligned} P(X_{ij1}) &= P(X_{ij1} = 1) \\ &\dots \\ P(X_{ijk}) &= \frac{P(X_{ij} = k)}{\prod_{l=1}^{k-1} (1 - P(X_{ijl}))} \end{aligned}$$

should be replaced by

The Boolean variables are associated with the following parameters:

$$\begin{aligned} P(X_{ij1}) &= P(X_{ij1} = 1) \\ &\dots \\ P(X_{ijk}) &= \frac{P(X_{ij} = k)}{\prod_{l=1}^{k-1} (1 - P(X_{ijl}))} \end{aligned}$$

Page 174

The text:

To define structured decomposability, consider a Deterministic Decomposable Negation Normal Form (d-DNNF) δ and assume, without loss of generality, that all conjunctions are binary. δ respects a vtree V if for every conjunction $\alpha \wedge \beta$ in δ , there is a node v in V such that $\text{vars}(\alpha) \subseteq \text{vars}(v_l)$ and $\text{vars}(\beta) \subseteq \text{vars}(v_r)$ where v_l and v_r are the left and right child of v . δ enjoys *structured decomposability* if it satisfies some vtree.

should be replaced by

To define structured decomposability, consider a d-DNNF δ and assume, without loss of generality, that all conjunctions are binary. δ respects a vtree V if for every conjunction $\alpha \wedge \beta$ in δ , there is a node v in V such that $\text{vars}(\alpha) \subseteq \text{vars}(v_l)$ and $\text{vars}(\beta) \subseteq \text{vars}(v_r)$ where v_l and v_r are the left and right child of v and $\text{vars}(v)$ is the set of variables appearing in d-DNNF v . δ enjoys *structured decomposability* if it satisfies some vtree.

Page 176, Definition 35

The text:

Definition 35 (T_{CP} operator [Vlasselaer et al., 2015, 2016]). *Let \mathcal{P} be a ground probabilistic logic program with probabilistic facts \mathcal{F} and atoms $\mathcal{B}_{\mathcal{P}}$. Let \mathcal{I} be a parameterized interpretation with pairs (a, λ_a) . Then, the T_{CP} operator is $T_{CP}(\mathcal{I}) = \{(a, \lambda_a) | a \in \mathcal{B}_{\mathcal{P}}\}$ where*

$$\lambda'_a = \begin{cases} a & \text{if } a \in \mathcal{F} \\ \bigvee_{a \leftarrow b_1, \dots, b_n, \sim e_1, \dots, \sim e_m \in \mathcal{R}} (\lambda_{b_1} \wedge \dots \wedge \lambda_{b_n} \wedge \neg \lambda_{c_1} \wedge \dots \wedge \neg \lambda_{c_m}) & \text{if } a \in \mathcal{B}_{\mathcal{P}} \setminus \mathcal{F} \end{cases}$$

should be replaced by

Definition 35 (T_{CP} operator [Vlasselaer et al., 2015, 2016]). *Let \mathcal{P} be a ground probabilistic logic program with probabilistic facts \mathcal{F} , rules \mathcal{R} and atoms $\mathcal{B}_{\mathcal{P}}$. Let \mathcal{I} be a parameterized interpretation with pairs (a, λ_a) . Then, the T_{CP} operator is $T_{CP}(\mathcal{I}) = \{(a, \lambda_a) | a \in \mathcal{B}_{\mathcal{P}}\}$ where*

$$\lambda'_a = \begin{cases} a & \text{if } a \in \mathcal{F} \\ \bigvee_{a \leftarrow b_1, \dots, b_n, \sim e_1, \dots, \sim e_m \in \mathcal{R}} (\lambda_{b_1} \wedge \dots \wedge \lambda_{b_n} \wedge \neg \lambda_{c_1} \wedge \dots \wedge \neg \lambda_{c_m}) & \text{if } a \in \mathcal{B}_{\mathcal{P}} \setminus \mathcal{F} \end{cases}$$

Page 177

The text:

Vlasselaer et al. [2016] show that if each atom is selected frequently enough in step 1, then the same fixpoint $\text{lfp}(T_{CP})$ is reached as for the naive algorithm, provided that the operator is still applied stratum by stratum in normal logic programs.

should be replaced by

Vlasselaer et al. [2016] show that if each atom is selected frequently enough in step 1, then the same fixpoint $\text{lfp}(T_{CP})$ is reached as for the naive algorithm that considers all atoms at the same time, provided that the operator is still applied stratum by stratum in normal logic programs.

Page 194

Formula:

$$\begin{aligned}\psi_{g_1}(X) &= \mathbb{M}(\psi''_{g_2}, Z) = \int_Z \psi''_{g_2} \downarrow_Z = \\ &= 0.3\mathcal{N}_Z(2.5, 1.1) + 0.7\mathcal{N}_X(3.5, 1.1)\end{aligned}$$

should be replaced by

$$\begin{aligned}\psi_{g_1}(X) &= \mathbb{M}(\psi''_{g_2}, Z) = \int_Z \psi''_{g_2} \downarrow_Z = \\ &= 0.3\mathcal{N}_X(2.5, 1.1) + 0.7\mathcal{N}_X(3.5, 1.1)\end{aligned}$$

Page 247, Algorithm 11

The text:

Algorithm 11 Function EXACTSOLUTION: Solving the DTPROBLOG decision problem exactly.

```

1: function EXACTSOLUTION( $\mathcal{DT}$ )
2:    $ADD_{tot}^{util} \leftarrow 0$ 
3:   for all  $(u \rightarrow r) \in \mathcal{U}$  do
4:     Build  $BDD(u)$ , the BDD for  $u$ 
5:      $ADD(u) \leftarrow \text{PROBABILITYDD}(BDD_u(\mathcal{DT}))$ 
6:      $ADD^{util}(u) \leftarrow r \cdot ADD_u(\sigma)$ 
7:      $ADD_{tot}^{util} \leftarrow ADD_{tot}^{util} \oplus ADD^{util}(u)$ 
8:   end for
9:   let  $t_{max}$  be the terminal node of  $ADD_{tot}^{util}$  with the highest utility
10:  let  $p$  be a path from  $t_{max}$  to the root of  $ADD_{tot}^{util}$ 
11:  return the Boolean decisions made on  $p$ 
12: end function
13: function PROBABILITYDD( $n$ )
14:   if  $n$  is the 1-terminal then
15:     return a 1-terminal
16:   end if
17:   if  $n$  is the 0-terminal then
18:     return a 0-terminal
19:   end if
20:   let  $h$  and  $l$  be the high and low children of  $n$ 
21:    $ADD_h \leftarrow \text{PROBABILITYDD}(h)$ 
22:    $ADD_l \leftarrow \text{PROBABILITYDD}(l)$ 
23:   if  $n$  represents a decision  $d$  then
24:     return  $\text{ITE}(d, ADD_h, ADD_l)$ 
25:   end if
26:   if  $n$  represents a fact with probability  $p$  then
27:     return  $(p \cdot ADD_h) \oplus ((1 - p) \cdot ADD_l)$ 
28:   end if
29: end function

```

should be replaced by

Algorithm 11 Function EXACTSOLUTION: Solving the DTPROBLOG decision problem exactly.

```

1: function EXACTSOLUTION( $\mathcal{DT}$ )
2:    $ADD_{tot}^{util} \leftarrow 0$ 
3:   for all  $(u \rightarrow r) \in \mathcal{U}$  do
4:     Build BDD( $u$ ), the BDD for  $u$ 
5:      $ADD(u) \leftarrow$  PROBABILITYDD(BDD( $u$ ))
6:      $ADD^{util}(u) \leftarrow r \cdot ADD(u)$ 
7:      $ADD_{tot}^{util} \leftarrow ADD_{tot}^{util} \oplus ADD^{util}(u)$ 
8:   end for
9:   let  $t_{max}$  be the terminal node of  $ADD_{tot}^{util}$  with the highest utility
10:  let  $p$  be a path from  $t_{max}$  to the root of  $ADD_{tot}^{util}$ 
11:  return the Boolean decisions made on  $p$ 
12: end function
13: function PROBABILITYDD( $n$ )
14:  if  $n$  is the 1-terminal then
15:    return a 1-terminal
16:  end if
17:  if  $n$  is the 0-terminal then
18:    return a 0-terminal
19:  end if
20:  let  $h$  and  $l$  be the high and low children of  $n$ 
21:   $ADD_h \leftarrow$  PROBABILITYDD( $h$ )
22:   $ADD_l \leftarrow$  PROBABILITYDD( $l$ )
23:  if  $n$  represents a decision  $d$  then
24:    return ITE( $d$ ,  $ADD_h$ ,  $ADD_l$ )
25:  end if
26:  if  $n$  represents a fact with probability  $p$  then
27:    return  $(p \cdot ADD_h) \oplus ((1 - p) \cdot ADD_l)$ 
28:  end if
29: end function

```

Pages 260-261

The text:

To perform Expectation Maximization (EM), we can associate a random variable X_{ij} with values $D = \{x_{i1}, \dots, x_{in_i}\}$ to the ground switch name $i\theta_j$ of $msw(i, x)$ with domain D , with θ_j being a grounding substitution for i . Let $g(i)$ be the set of such substitutions:

$$g(i) = \{j|\theta_j \text{ is a grounding substitution for } i \text{ in } msw(i, x)\}.$$

The EM algorithm alternates between the two phases:

- Expectation: computes $E[c_{ik}|e]$ for all examples e , switches $msw(i, x)$ and $k \in \{1, \dots, n_i\}$, where c_{ik} is the number of times a variable X_{ij}

takes value x_{ik} with j in $g(i)$. $\mathbf{E}[c_{ik}|e]$ is given by $\sum_{j \in g(i)} P(X_{ij} = x|e)$.

- Maximization: computes Π_{ik} for all $msw(i, x)$ and $k = 1, \dots, n_i - 1$ as

$$\Pi_{ik} = \frac{\sum_{e \in E} \mathbf{E}[c_{ik}|e]}{\sum_{e \in E} \sum_{k=1}^{n_i} \mathbf{E}[c_{ik}|e]}$$

So, for each example e , X_{ij} s and x_{ik} s, we compute $P(X_{ij} = x_{ik}|e)$, the expected value of X_{ij} given the example, with $k \in \{1, \dots, n_i\}$. These expected values are then aggregated and used to complete the dataset for computing the parameters by relative frequency. If c_{ik} is number of times a variable X_{ij} takes value x_{ik} for any j , $\mathbf{E}[c_{ik}|e]$ is its expected value given example e . if $\mathbf{E}[c_{ik}]$ is its expected value given all the examples, then

$$\mathbf{E}[c_{ik}] = \sum_{t=1}^T \mathbf{E}[c_{ik}|e_t]$$

and

$$\Pi_{ik} = \frac{\mathbf{E}[c_{ik}]}{\sum_{k=1}^{n_i} \mathbf{E}[c_{ik}]}$$

should be replaced by

To perform EM, we can associate a random variable X_{ij} with values $D = \{x_{i1}, \dots, x_{in_i}\}$ to the ground switch name $i\theta_j$ of $msw(i, x)$ with domain D , with θ_j being a grounding substitution for i . Let $g(i)$ be the set of such substitutions:

$$g(i) = \{j|\theta_j \text{ is a grounding substitution for } i \text{ in } msw(i, x)\}.$$

PRISM will learn different parameters for each X_{ij} random variable. The EM algorithm alternates between the two phases:

- Expectation: computes $\mathbf{E}[c_{ijk}|e]$ for all examples e , switches $msw(i\theta_j, x)$ and $k \in \{1, \dots, n_i\}$, where c_{ijk} is the number of times variable X_{ij} takes value x_{ik} . $\mathbf{E}[c_{ijk}|e]$ is given by $P(X_{ij} = x_{ik}|e)$.
- Maximization: computes Π_{ijk} for all $msw(i\theta_j, x)$ and $k = 1, \dots, n_i - 1$ as

$$\Pi_{ijk} = \frac{\sum_{e \in E} \mathbf{E}[c_{ijk}|e]}{\sum_{e \in E} \sum_{k=1}^{n_i} \mathbf{E}[c_{ijk}|e]}$$

So, for each example e , X_{ij} s and x_{ik} s, we compute $P(X_{ij} = x_{ik}|e)$, the expected value of X_{ij} given the example, with $k \in \{1, \dots, n_i\}$. These expected values are then used to complete the dataset for computing the parameters by relative frequency. If c_{ijk} is number of times a variable X_{ij} takes value x_{ik} , $\mathbf{E}[c_{ijk}|e]$ is its expected value given example e . If $\mathbf{E}[c_{ijk}]$ is its expected value given all the examples, then

$$\mathbf{E}[c_{ijk}] = \sum_{t=1}^T \mathbf{E}[c_{ijk}|e_t]$$

and

$$\Pi_{ijk} = \frac{\mathbf{E}[c_{ijk}]}{\sum_{k=1}^{n_i} \mathbf{E}[c_{ijk}]}.$$

Page 262-263

The text:

Algorithm 13 Function PRISM-EM: Naive EM learning in PRISM

```

1: function PRISM-EM-NAIVE( $E, \mathcal{P}, \epsilon$ )
2:    $LL = -inf$ 
3:   repeat
4:      $LL_0 = LL$ 
5:     for all  $i, k$  do                                     ▷ Expectation step
6:        $\mathbf{E}[c_{ik}] \leftarrow \sum_{e \in E} \frac{\sum_{\kappa \in \mathcal{K}_e, ms_w(i, x_{ik}) \theta_j \in e} P(\kappa)}{P(e)}$ 
7:     end for
8:     for all  $i, k$  do                                     ▷ Maximization step
9:        $\Pi_{ik} \leftarrow \frac{\mathbf{E}[c_{ik}]}{\sum_{k'=1}^{n_i} \mathbf{E}[c_{ik'}]}$ 
10:    end for
11:     $LL \leftarrow \sum_{e \in E} \log P(e)$ 
12:  until  $LL - LL_0 < \epsilon$ 
13:  return  $LL, \Pi_{ik}$  for all  $i, k$ 
14: end function

```

Algorithm 14 Procedure GET-INSIDE-PROBS: computation of inside probabilities.

```

1: procedure GET-INSIDE-PROBS( $q$ )
2:   for all  $i, k$  do
3:      $P(ms_w(i, v_k)) \leftarrow \Pi_{ik}$ 
4:   end for
5:   for  $i \leftarrow m \rightarrow 1$  do
6:      $P(g_i) \leftarrow 0$ 
7:     for  $j \leftarrow 1 \rightarrow s_i$  do
8:       Let  $S_{ij}$  be  $h_{ij1}, \dots, h_{ij s_i}$ 
9:        $P(g_i, S_{ij}) \leftarrow \prod_{l=1}^{s_i} P(h_{ijl})$ 
10:       $P(g_i) \leftarrow P(g_i) + P(g_i, S_{ij})$ 
11:    end for
12:  end for
13: end procedure

```

should be replaced by

Algorithm 13 Function PRISM-EM-NAIVE: Naive EM learning in PRISM

```
1: function PRISM-EM-NAIVE( $E, \mathcal{P}, \epsilon$ )
2:    $LL = -inf$ 
3:   repeat
4:      $LL_0 = LL$ 
5:     for all  $i, j, k$  do ▷ Expectation step
6:        $\mathbf{E}[c_{ijk}] \leftarrow \sum_{e \in E} \frac{\sum_{\kappa \in K_e, ms_w(i\theta_j, x_{ik}) \in e} P(\kappa)}{P(e)}$ 
7:     end for
8:     for all  $i, j, k$  do ▷ Maximization step
9:        $\Pi_{ijk} \leftarrow \frac{\mathbf{E}[c_{ijk}]}{\sum_{k'=1}^{n_i} \mathbf{E}[c_{ijk'}]}$ 
10:    end for
11:     $LL \leftarrow \sum_{e \in E} \log P(e)$ 
12:  until  $LL - LL_0 < \epsilon$ 
13:  return  $LL, \Pi_{ijk}$  for all  $i, j, k$ 
14: end function
```

Algorithm 14 Procedure GET-INSIDE-PROBS: computation of inside probabilities.

```
1: procedure GET-INSIDE-PROBS( $e$ )
2:   for all  $i, j, k$  do
3:      $P(ms_w(i\theta_j, v_k)) \leftarrow \Pi_{ijk}$ 
4:   end for
5:   for  $i \leftarrow m \rightarrow 1$  do
6:      $P(g_i) \leftarrow 0$ 
7:     for  $j \leftarrow 1 \rightarrow s_i$  do
8:       Let  $S_{ij}$  be  $h_{ij1}, \dots, h_{ijo}$ 
9:        $P(g_i, S_{ij}) \leftarrow \prod_{l=1}^o P(h_{ijl})$ 
10:       $P(g_i) \leftarrow P(g_i) + P(g_i, S_{ij})$ 
11:    end for
12:  end for
13: end procedure
```

The text

Outside probabilities instead are defined as

$$Q(g_i) = \frac{\partial P(q)}{\partial P(g_i)}$$

should be replaced by

Outside probabilities instead are defined as

$$Q(g_i) = \frac{\partial P(e)}{\partial P(g_i)}$$

Pages 263-264

The text

We have that $Q(g_1) = 1$ as $q = g_1$. For $i = 2, \dots, m$, we can derive $Q(g_i)$ by the chain rule of the derivative knowing that $P(q)$ is a function of $P(b_1), \dots, P(b_K)$

$$Q(g_i) = \frac{\partial P(q)}{\partial P(b_1)} \frac{\partial P(g_i, W_{11})}{\partial P(g_1)} + \dots + \frac{\partial P(q)}{\partial P(b_K)} \frac{\partial P(g_i, W_{K i_K})}{\partial P(g_1)} = Q(b_1)P(g_i, W_{11})/P(g_i) + \dots + Q(b_K)P(g_i, W_{K i_K})/P(g_i)$$

should be replaced by

We have that $Q(g_1) = 1$ as $e = g_1$. For $i = 2, \dots, m$, we can derive $Q(g_i)$ by the chain rule of the derivative knowing that $P(e)$ is a function of $P(b_1), \dots, P(b_K)$

$$Q(g_i) = \frac{\partial P(e)}{\partial P(b_1)} \frac{\partial P(g_i, W_{11})}{\partial P(g_1)} + \dots + \frac{\partial P(e)}{\partial P(b_K)} \frac{\partial P(g_i, W_{K i_K})}{\partial P(g_1)} = Q(b_1)P(g_i, W_{11})/P(g_i) + \dots + Q(b_K)P(g_i, W_{K i_K})/P(g_i)$$

The text:

If $g_i = msw(i, x_k)\theta_j$, then

$$P(X_{ij} = x_{ik}, e) = Q(g_i)P(g_i) = Q(g_i)\Pi_{ik}.$$

In fact, we can divide the explanations for e into two sets, K_{e1} , that includes the explanations containing $msw(i, x_k)\theta_j$, and K_{e2} , that includes the other explanations. Then $P(e) = P(K_{e1}) + P(K_{e2})$ and $P(X_{ij} = x_{ik}, e) = P(K_{e1})$. Since each explanation in K_{e1} contains $g_i = msw(i, x_k)\theta_j$, K_{e1} takes the form $\{\{g_i, W_1\}, \dots, \{g_i, W_s\}\}$ and

should be replaced by

If $g_i = msw(i\theta_j, x_k)$, then

$$P(X_{ij} = x_{ik}, e) = Q(g_i)P(g_i) = Q(g_i)\Pi_{ijk}.$$

In fact, we can divide the explanations for e into two sets, K_{e1} , that includes the explanations containing $msw(i\theta_j, x_k)$, and K_{e2} , that includes the other explanations. Then $P(e) = P(K_{e1}) + P(K_{e2})$ and $P(X_{ij} = x_{ik}, e) = P(K_{e1})$. Since each explanation in K_{e1} contains $g_i = msw(i\theta_j, x_k)$, K_{e1} takes the form $\{\{g_i, W_1\}, \dots, \{g_i, W_s\}\}$ and

The text:

Algorithm 15 Procedure GET-OUTSIDE-PROBS: computation of outside probabilities.

```
1: procedure GET-OUTSIDE-PROBS( $q$ )
2:    $Q(g_1) \leftarrow 1.0$ 
3:   for  $i \leftarrow 2 \rightarrow m$  do
4:      $Q(g_i) \leftarrow 0.0$ 
5:   end for
6:   for  $i \leftarrow 2 \rightarrow m$  do
7:      $Q(g_i) \leftarrow 0.0$ 
8:     for  $j \leftarrow 1 \rightarrow s_i$  do
9:       Let  $S_{ij}$  be  $h_{ij1}, \dots, h_{ijo}$ 
10:      for  $l \leftarrow 1 \rightarrow o$  do
11:         $Q(h_l) \leftarrow Q(h_l) + Q(g_i)P(g_i, S_{ij})/P(h_{ijl})$ 
12:      end for
13:    end for
14:  end for
15: end procedure
```

should be replaced by

Algorithm 15 Procedure GET-OUTSIDE-PROBS: computation of outside probabilities.

```
1: procedure GET-OUTSIDE-PROBS( $e$ )
2:    $Q(g_1) \leftarrow 1.0$ 
3:   for  $i \leftarrow 2 \rightarrow m$  do
4:      $Q(g_i) \leftarrow 0.0$ 
5:   end for
6:   for  $i \leftarrow 2 \rightarrow m$  do
7:      $Q(g_i) \leftarrow 0.0$ 
8:     for  $j \leftarrow 1 \rightarrow s_i$  do
9:       Let  $S_{ij}$  be  $h_{ij1}, \dots, h_{ijo}$ 
10:      for  $l \leftarrow 1 \rightarrow o$  do
11:         $Q(h_l) \leftarrow Q(h_l) + Q(g_i)P(g_i, S_{ij})/P(h_{ijl})$ 
12:      end for
13:    end for
14:  end for
15: end procedure
```

Page 265, Algorithms 16-17

The text:

Algorithm 16 Function PRISM-EM

```
1: function PRISM-EM( $E, \mathcal{P}, \epsilon$ )
2:    $LL = -inf$ 
3:   repeat
4:      $LL_0 = LL$ 
5:      $LL = \text{EXPECTATION}(E)$ 
6:     for all  $i$  do
7:        $Sum \leftarrow \sum_{k=1}^{n_i} \mathbf{E}[c_{ik}]$ 
8:       for  $k = 1$  to  $n_i$  do
9:          $\Pi_{ik} = \frac{\mathbf{E}[c_{ik}]}{Sum}$ 
10:      end for
11:    end for
12:  until  $LL - LL_0 < \epsilon$ 
13:  return  $LL, \Pi_{ik}$  for all  $i, k$ 
14: end function
```

Algorithm 17 Procedure PRISM-EXPECTATION

```
1: function PRISM-EXPECTATION( $E$ )
2:    $LL = 0$ 
3:   for all  $e \in E$  do
4:     GET-INSIDE-PROBS( $e$ )
5:     GET-OUTSIDE-PROBS( $e$ )
6:     for all  $i$  do
7:       for  $k = 1$  to  $n_i$  do
8:          $\mathbf{E}[c_{ik}] = \mathbf{E}[c_{ik}] + Q(msw(i, x_k))\Pi_{ik}/P(e)$ 
9:       end for
10:    end for
11:     $LL = LL + \log P(e)$ 
12:  end for
13:  return  $LL$ 
14: end function
```

should be replaced by

Algorithm 16 Function PRISM-EM

```
1: function PRISM-EM( $E, \mathcal{P}, \epsilon$ )
2:    $LL = -inf$ 
3:   repeat
4:      $LL_0 = LL$ 
5:      $LL = \text{PRISM-EXPECTATION}(E)$ 
6:     for all  $i, j$  do
7:        $Sum \leftarrow \sum_{k=1}^{n_i} \mathbf{E}[c_{ijk}]$ 
8:       for  $k = 1$  to  $n_i$  do
9:          $\Pi_{ijk} = \frac{\mathbf{E}[c_{ijk}]}{Sum}$ 
10:      end for
11:    end for
12:  until  $LL - LL_0 < \epsilon$ 
13:  return  $LL, \Pi_{ijk}$  for all  $i, j, k$ 
14: end function
```

Algorithm 17 Procedure PRISM-EXPECTATION

```
1: function PRISM-EXPECTATION( $E$ )
2:    $LL = 0$ 
3:   for all  $e \in E$  do
4:     GET-INSIDE-PROBS( $e$ )
5:     GET-OUTSIDE-PROBS( $e$ )
6:     for all  $i, j$  do
7:       for  $k = 1$  to  $n_i$  do
8:          $\mathbf{E}[c_{ijk}] = \mathbf{E}[c_{ijk}] + Q(msw(i\theta_j, x_k))\Pi_{ijk}/P(e)$ 
9:       end for
10:    end for
11:     $LL = LL + \log P(e)$ 
12:  end for
13:  return  $LL$ 
14: end function
```

Page 272

The text:

$$\pi_{ik} = \frac{\sum_{e \in E} \mathbf{E}[c_{ik1}|e]}{\sum_{q \in E} \mathbf{E}[c_{ik0}|e] + \mathbf{E}[c_{ik1}|e]}$$

should be replaced by

$$\pi_{ik} = \frac{\sum_{e \in E} \mathbf{E}[c_{ik1}|e]}{\sum_{e \in E} \mathbf{E}[c_{ik0}|e] + \mathbf{E}[c_{ik1}|e]}$$

Page 275

The two formulas should be replaced by

$$P(X_{ijk} = 0, e) = \sum_{n \in N(X_{ijk})} e^0(n) + (1 - \pi_{ik}) \left(\sum_{n \in Del^0(X_{ijk})} e^0(n) + \sum_{n \in Del^1(X_{ijk})} e^1(n) \right)$$
$$P(X_{ijk} = 1, e) = \sum_{n \in N(X_{ijk})} e^1(n) + \pi_{ik} \left(\sum_{n \in Del^0(X_{ijk})} e^0(n) + \sum_{n \in Del^1(X_{ijk})} e^1(n) \right)$$

Page 276

The formula

$$\frac{\partial P(f(\mathbf{X}))}{\partial \Pi_j} = \Pi_k \cdot \frac{\partial P(f^{X_k}(\mathbf{X}))}{\partial \Pi_j} + (1 - \Pi_k) \cdot \frac{\partial P(f^{-X_k}(\mathbf{X}))}{\partial \Pi_j}$$

should be replaced by

$$\frac{\partial P(f(\mathbf{X}))}{\partial \Pi_j} = \Pi_k \cdot \frac{\partial P(f^{X_k}(\mathbf{X}))}{\partial \Pi_j} + (1 - \Pi_k) \cdot \frac{\partial P(f^{-X_k}(\mathbf{X}))}{\partial \Pi_j}$$

Page 277 Algorithm 22

$\pi(ik)$ should be replaced by π_{ik} in line 4.

Page 278 Algorithm 24

η_t^0 and η_t^1 in lines 10 and 11 should be replaced respectively by η^0 and η^1 .

Page 281

The text:

LFI-ProbLog computes $P(X_{ij} = x|\mathcal{I})$ by computing $P(X_{ij} = x, \mathcal{I})$ using Procedure CIRCP shown in Algorithm 5: the d-DNNF circuit is visited twice, once bottom up to compute $P(q(\mathcal{I}))$ and once top down to compute $P(X_{ij} = x|\mathcal{I})$ for all the variables X_{ij} and values x . Then $P(X_{ij} = x|\mathcal{I})$ is given by $\frac{P(X_{ij}=x, \mathcal{I})}{P(\mathcal{I})}$.

should be replaced by

LFI-ProbLog computes $P(X_{ij} = x|\mathcal{I})$ by computing $P(X_{ij} = x, \mathcal{I})$ using Procedure CIRCP shown in Algorithm 5: the d-DNNF circuit is visited twice, once bottom up to compute $P(q(\mathcal{I}))$ and once top down to compute $P(X_{ij} = x, \mathcal{I})$ for all the variables X_{ij} and values x . Then $P(X_{ij} = x|\mathcal{I})$ is given by $\frac{P(X_{ij}=x, \mathcal{I})}{P(\mathcal{I})}$.

Page 292 Algorithm 26

target \leftarrow *bod, refinement* in line 18 should be replaced with *target* \leftarrow *body, refinement*.

References

- J. Vlasselaer, G. Van den Broeck, A. Kimmig, W. Meert, and L. De Raedt. Anytime inference in probabilistic logic programs with Tp-compilation. In *24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1852–1858, 2015.
- J. Vlasselaer, G. Van den Broeck, A. Kimmig, W. Meert, and L. De Raedt. Tp-compilation for inference in probabilistic logic programs. *International Journal of Approximate Reasoning*, 78:15–32, 2016. doi: 10.1016/j.ijar.2016.06.009.